Efficient Multiple Change Point Detection in Time-varying Markov Random Fields

Qilong Ding*
Department of Statistics and Actuarial Science, The University of Hong Kong
and
Bin Liu
Department of Statistics and Data Science, School of Management, Fudan University

January 17, 2025

Abstract

This study focuses primarily on estimating multiple change points within a time-varying binary Markov random field that evolves piecewise continuously. In dynamic network structures, change points signify crucial moments when significant alterations occur, therefore identifying the change points is of vital meaning. To achieve this, we employ a pseudo-likelihood function with an L_1 penalty to introduce a joint estimator for both the change points and the underlying graph structures. The optimization process can be carried out efficiently using dynamic programming and binary segmentation techniques. Furthermore, we demonstrate that the objective function of pseudo-likelihood can be equivalently transformed into the loss function of logistic regression while reserving the time order, enhancing the accuracy of graph structure estimation and the efficiency of the change point grid search. Lastly, we assess the performance of our strategies using simulation data and compare them with the baseline method. We also apply them to the voting data and futures data, explore the effectiveness in the real-world modeling, and gain some insightful perspectives.

Keywords: Markov Random Field, Change point detection, Pseudo-likelihood, Logistic Regression, Network analysis.

*The authors gratefully acknowledge *please remember to list all relevant funding sources in the unblinded version*

1 Introduction

Graphs are fundamental tools for modeling and studying static or dynamic relationships among latent high-dimensional vector data. There are numerous applications in physics, computer vision, and statistics. In static scenarios, the learning of relationships between variables is referred to as graph inference. Markov networks are a class of classical probabilistic graphical models widely used in various fields such as image processing, multiple testing, and computational biology. In Markov networks, random variables are represented as different nodes in the graph, where edges between two nodes in the graph indicate conditional dependencies between the corresponding random variables given other variables in Koller et al. (2009).

The learning of the structure of Markov Random Fields (MRFs) from a set of observed outcomes has been widely researched (Banerjee et al., 2008; Meinshausen et al., 2006), especially in the case of Gaussian graphical models (Yuan and Lin, 2007) and the well-known graphical lasso (Friedman et al., 2008). Inference tasks for binary Markov Random Field models, also known as the Ising Model, have also been addressed in the past (Höfling and Tibshirani, 2009; Ravikumar et al., 2010; Xue et al., 2012).

However, previous methods have assumed that the underlying structure remains constant, with all observed samples conforming to the same underlying structure (referred to as static graph models), without considering the case of evolving underlying structures over time. On a temporal scale, the internal structure of graphical models may change over time, which we refer to as time-varying graph models. In recent years, there has been a strong interest in learning the structure of time-varying graph models (Hallac et al., 2017; Yang and Peng, 2019). This task can be combined with change point detection in time series, aiming to detect significant moments when the graph structure undergoes noticeable changes. These prominent changes may be due to endogenous evolution or exogenous impacts of events, and identifying temporal change points is of vital importance

in characterizing the evolution of graph models and identifying critical moments, making it particularly intriguing. For piecewise constant Gaussian graphical models (Gibberd and Nelson, 2017; Wang et al., 2018; Londschien et al., 2019), extensive research has been conducted in various types of change point detection objectives, such as single change point (Bybee and Atchadé, 2018), multiple change points (Gibberd and Roy, 2017), offline detection (Kolar and Xing, 2012; Gibberd and Nelson, 2017), online detection (Keshavarz et al., 2018), and more.

Up to now, there have been limited advancements related to time-varying Markov Random Field (MRF) models. Particularly, the combination of multiple change point detection and structural inference has not been adequately explored in the past. According to Ahmed and Xing (2009); Kolar et al. (2010), the authors focused on learning the parameters of time-varying Ising models without the need to identify crucial change points since the network can change at every time point. In Fazayeli and Banerjee (2016), the authors assumed that change point locations were known and concentrated solely on inferring structural changes between Ising models. More recently, Roy et al. (2017) studied the problem of detecting a single change point. Based on the aforementioned, this paper primarily addresses the issue of detecting multiple change points in time-varying Markov Random Fields.

It is worth noting that the multiple change-point detection method developed in this paper has a wide range of applications, closely related to the applications of static Markov Random Fields. For instance, in biological environments, regulatory networks composed of genes dynamically change over the course of treatment time, exhibiting significant alterations under specific drug dosages. Similarly, in financial markets, economic networks evolve into a series of temporal networks over time, where sudden external shocks or major economic announcements can disrupt the financial network at any given moment, thereby affecting the internal structure of the network. It is important to distinguish our work here from the content in Kolar and Xing (2012), which considers Gaussian graphical models

where each node in the graph has a corresponding change point. In contrast, our focus is on the temporal change points that impact the global network structure of the underlying Markov Random Field. This difference in focus arises from distinct application scenarios. In biological applications that concentrate on specific biomolecules such as genes, proteins, or metabolites, analyzing change points at the granularity of network nodes is typically prioritized. On the other hand, in many social network applications like financial economic networks in empirical studies, changes in the global structure of the network are the primary concern. Furthermore, it's important to note that due to varying importance of each node, the extent of change at the node level can vary, resulting in "inconsistent" changes that may be challenging to reconcile. These node-level changes may also differ significantly from the global structural change points that this paper focuses on.

1.1 Contributions and Paper Organizations

In this paper, we consider data generated from the Ising model, which is a binary Markov Random Field. It is a probabilistic undirected graphical model where each node takes values from $\{0, 1\}$, and the edges in the network represent pairwise dependencies (more precisely, conditional dependencies) among nodes. While static Markov Random Fields have a likelihood function, estimating binary Markov networks has been tricky. The most persistent challenge has been dealing with computationally intricate calculations associated with the log-likelihood, making the process of graphical inference an NP-hard problem. Therefore, in the literature, various methods have been proposed to approximate the log-likelihood rather than using exact estimates. Hinton (2002) established the contrastive divergence algorithm by directly estimating the log-likelihood derivatives of discrete Markov networks. Ravikumar et al. (2010) separately considered neighborhood recovery for each variable and introduced the node-wise logistic regression (NLR) approach. In Höfling and Tibshirani (2009), pseudo-likelihood was introduced as an approximation to the log-likelihood of

Markov networks with penalty terms.

According to Höfling and Tibshirani (2009), the Pseudo-Likelihood (PL) method is one of the most competitive methods, offering faster speed and higher accuracy compared to other methods, encapsulated in the R software package `BMN` (Höfling and Tibshirani, 2009). However, due to advances in optimization theory, the `BMN` method has become one of the slowest among many learning methods for discrete Markov networks (Viallon et al., 2014). Additionally, due to the specific nature of the `BMN` implementation algorithm, a large number of samples are required for learning parameters, resulting in a high sample cost for estimation. Therefore, a more efficient PL optimization process is still needed. As some researchers have noted, there are underlying similarities between the objective functions of PL and Logistic Regression (LR). For example, Höfling and Tibshirani (2009) points out that the PL model is related to the LR problem. Yang and Ravikumar (2011) suggests that PL is a LR problem with symmetric constraints. At the same time, L_1-regularized logistic regression (LR), as one of the most widely used generalized linear models, has a complex implementation but can efficiently provide solutions. The implementation of L_1-regularized generalized linear models utilizes coordinate descent (Friedman et al., 2010) and variable selection (Tibshirani et al., 2012) and has become a building block for many other sparse learning problems, such as Zhao et al. (2018). Therefore, inspired by the relationship between PL and LR, we shall transform the static PL estimation problem into an LR parameter estimation problem and utilize the optimization tools of L_1-regularized generalized linear models to achieve fast optimization of the PL problem, which has also been shown to be a more accurate solution within the same sample size compared to `BMN`.

Besides introducing the commonly used change point search algorithms into our special data types, we innovatively improves the estimation methods on Markov random fields through the equivalence transformation into logistic regression. It will greatly enhance the efficiency and accuracy of the whole multiple change point detection framework, which is

elaborated in our algorithm analysis and verified in the numerical experiment.

After our discussion, we have gained a thorough understanding of the Markov field data type. Now, based on this foundation, we consider the problem of change point detection, particularly multiple change points. We initially consider the detection of a single change point in Markov Random Fields. We randomly divide a segment of the Markov Random Field over a period of time into two segments and calculate the whole likelihood of these two segments through parameter estimation. The split point which maximizes the sum of the likelihood functions of the two segments is considered as the so-called change point. If we view the total likelihood as a form of loss function for a specific segmentation, by minimizing this loss function, we can find the potential connections of the multi-change point problem for this type of data with multiple change point problems for other data types. Actually, this idea comes from the extensive literature on multiple change point problems for other data types. For instance, Leonardi and Bühlmann (2016) proposed effective detection of multiple change points in high-dimensional regression models. They utilized the commonly used sum of squares loss in regression as the loss function for segmentation, applied dynamic programming and binary segmentation to minimize the loss, and obtained the final change point locations and parameter estimates for each segment of time.

The structure of this article is organized as follows. In Section 2 we introduce dynamic programming and binary segmentation algorithms into time-varying Markov Random Fields, using the negative log-pseudo-likelihood function as the optimization objective (loss function), and minimize the loss function to obtain the final change point inferences and parameter estimates for each segment. It is worth noting that we employ the Pseudo-Likelihood as the surrogate of the likelihood for the consideration of intractable computation. During parameter estimation, we innovatively propose and prove the equivalence transformation between the PL method and logistic regression in Section 3. We compare it with the existing BMN method to highlight its advantages. Finally, in Section 4 and 5

we generate simulated data to validate the effectiveness of the algorithm and the novelty of the method through numerical experiments. We also perform empirical analysis using real data to demonstrate the applicability of the time-varying Markov Random Field model and the effectiveness of our developed change point detection algorithm.

2 Methodology

In this section, we primarily introduce what a time-varying Markov Random Field is, how to construct the model, and a series of algorithms we have designed for solving this model which is actually an optimization problem. We begin by extending the definition of a static Markov Random Field to the time series scale, thus obtaining the likelihood function that characterizes the time-varying Markov Random Field. Next, we introduce the Pseudo-Likelihood method as a robust approximation to the likelihood function. We incorporate the change point hypothesis and propose a joint estimation approach for both the change points and the internal structure of the time-varying Markov Random Field. Finally, we employ two algorithms, dynamic programming and binary segmentation, to compute the estimates for these parameters.

2.1 The time-varying binary Markov Random Field

We begin by introducing the static Markov Random Field, which is also an Ising model characterized by binary output features. This model can be defined by a class of graph structures denoted as $G = (V, E)$, where V represents the set of nodes, and E represents the set of edges (the connections between nodes). The edges connecting pairs of nodes represent their conditional independence, meaning that, given the other nodes, the linear dependency between these two nodes is completely characterized by the presence or absence of this edge. We define a p-dimensional binary random vector, denoted as $\boldsymbol{X} = (x_1, x_2, \cdots, x_p)^\top \in \{0, 1\}^p$, where all its components constitute the node set V. The edge set $E \subset V \times V$

can be expressed as a real symmetric matrix $\boldsymbol{\Theta} = [\theta_{ij}]_{p\times p}$. Then, we can get the following likelihood function of the random vector $\boldsymbol{X}$ which obeys the distribution of static Markov Random Field,

$$\mathbb{P}_{\boldsymbol{\Theta}}(\boldsymbol{X}) = \frac{1}{Z(\boldsymbol{\Theta})} \exp\left(\sum_{a\in V} \theta_{aa} x_a + \sum_{(a,b)\in E} \theta_{ab} x_a x_b\right), \tag{2.1}$$

where the term $Z(\boldsymbol{\Theta})$ is the corresponding normalizing constant defined as

$$Z(\boldsymbol{\Theta}) = \sum_{\boldsymbol{X}\in\{0,1\}^p} \exp\left(\sum_{a\in V} \theta_{aa} x_a + \sum_{(a,b)\in E} \theta_{ab} x_a x_b\right). \tag{2.2}$$

Based on this foundation, a time-varying Markov Random Field is defined as a time series of Markov Random Fields, denoted as $\{\boldsymbol{X}^{(i)}, 1 \le i \le T\}$, where the internal structural parameters vary over time, also denoted as $\{\boldsymbol{\Theta}^{(i)}, 1 \le i \le T\}$. The internal structural parameters of the Markov Random Field $\boldsymbol{X}^{(i)}$ at the i-th time point correspond to a real symmetric matrix $\boldsymbol{\Theta}^{(i)} \in \mathbb{R}^{p\times p}$ and have a likelihood function defined as in Equation (2.1). We assume that the internal structural parameters of the time-varying Markov Random Field are piecewise constant, and the observed samples $\{\boldsymbol{X}^{(i)}\}_{i=1}^{\top}$ are independently and identically distributed (i.i.d) within each segment. Then we assume that there exists a $(k+1)$-dimensional vector $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \cdots, \alpha_k)$, which satisfies that

$$0 = \alpha_0 < \alpha_1 < \cdots < \alpha_k = 1. \tag{2.3}$$

Corresponding to the segment vector α, there exist k real symmetric matrices $\{\boldsymbol{\Omega}^{(i)} \in \mathbb{R}^{p\times p}, 1 \le i \le k\}$, which satisfies that

$$\boldsymbol{\Theta}^{(i)} = \sum_{j=1}^{k} \boldsymbol{\Omega}^{(j)} \mathbb{I}\{i/T \in (\alpha_{j-1}, \alpha_j], 1 \le i \le T\}. \tag{2.4}$$

2.2 Surrogate for likelihood function

During parameter learning, computing the normalization constant $Z(\Theta)$ in the likelihood function shown in (2.1) has relatively high computational complexity. Therefore, classical maximum likelihood function maximization strategies are challenging to apply in the parameter learning of the static Markov Random Fields, let alone the time-Varying Markov

Random Fields. As a result, following previous research, we adopt the Pseudo-Likelihood method as the surrogate for the likelihood function (Höfling and Tibshirani, 2009), which is defined as follows:

Definition 2.1. Conditional likelihood for each node. For $\Theta \in \mathbb{R}^{p\times p}$ and any node $a \in V$, the node a have the conditional likelihood (given the other nodes) of the following form

$$\mathbb{P}_{\boldsymbol{\Theta_a}}(x_a|\boldsymbol{x_{-a}}) = \frac{1}{Z(\boldsymbol{\Theta_a})} \exp\left(\theta_{aa}x_a + \sum_{b\neq a} \theta_{ab}x_a x_b\right), \tag{2.5}$$

where $\boldsymbol{\Theta_a}$ represents the a-th column (or row) of Θ and $\boldsymbol{x_{-a}}$ denotes all the components of $\boldsymbol{X}_{V\setminus a}$. The normalizing constant $Z(\boldsymbol{\Theta_a})$ has a simpler form as follows,

$$Z(\boldsymbol{\Theta_a}) = 1 + \exp\left(\theta_{aa} + \sum_{b\neq a} \theta_{ab}x_b\right). \tag{2.6}$$

Definition 2.2. Pseudo-likelihood for a static Markov Random Field. The negative log pseudo-likelihood function of a static Markov Random Field is as follows,

$$\phi(\boldsymbol{\Theta}, \boldsymbol{X}) \coloneqq -\sum_{a=1}^{p} \log\left(\mathbb{P}_{\boldsymbol{\Theta_a}}(x_a|\boldsymbol{x_{-a}})\right). \tag{2.7}$$

It has been proven to be a good approximation of $-\log \mathbb{P}_{\boldsymbol{\Theta}}(\boldsymbol{X})$, and the strategy of minimizing the negative log pseudo-likelihood function is computationally more efficient than maximizing the original likelihood function.

2.3 Model setup

To simplify notations, without loss of generality, we assume that $\alpha_j T \in \mathbb{N}$ for all $j = 1, \ldots, k$. Given an interval $(u, v] \subset [0, 1]$ such that $uT, vT \in \mathbb{N}$, we denote by $\boldsymbol{X}^{(u,v]}$ the observations within the interval $\{\boldsymbol{X}^{(i)}, uT < i \leq vT\}$ and use $\boldsymbol{\Theta}^{(i)}$ to represent the associated real symmetric matrix at the time point i (also called graph structure). Analogously, we define the sum of negative log pseudo-likelihood within this interval as follows:

$$\Phi_{(u,v]} \coloneqq \sum_{i=uT+1}^{vT} \phi(\boldsymbol{\Theta}^{(i)}, \boldsymbol{X}^{(i)}). \tag{2.8}$$

It is worth noting that for every time point i, we adopt an individual $\boldsymbol{\Theta}^{(i)}$ without considering the piecewise constant model assumption, which we will apply the assumption to the model later on.

Furthermore, according to (2.3), we use $l(\boldsymbol{\alpha})$ to represent the count of positive components, denoted as $l(\boldsymbol{\alpha}) = k$. This value also corresponds to the number of segments in the time-varying model. For each $j = 1, \cdots, l(\boldsymbol{\alpha})$, we designate $I_j(\boldsymbol{\alpha})$ as the j-th interval in α and $r_j(\boldsymbol{\alpha})$ as its length; specifically, $I_j(\boldsymbol{\alpha}) = (\alpha_{j-1}, \alpha_j]$ and $r_j(\boldsymbol{\alpha}) = \alpha_j - \alpha_{j-1}$. Furthermore, we will use $r(\boldsymbol{\alpha})$ to denote the minimum size among such intervals, defined as

$$r(\boldsymbol{\alpha}) = \min_{j=1,\ldots,l(\boldsymbol{\alpha})} r_j(\boldsymbol{\alpha}). \tag{2.9}$$

In the sequel we will introduce $\|\cdot\|_r$ to represent the r-norm in $\mathbb{R}^p$. Let $\mathcal{M}_p$ be the space of all $p \times p$ real symmetric matrices. We equip $\mathcal{M}_p$ with the Frobenius inner product $\langle \boldsymbol{\Theta}, \boldsymbol{\Upsilon} \rangle := \sum_{j \le k} \theta_{jk} \upsilon_{jk}$, and the associated norm $\|\boldsymbol{\Theta}\|_F := \sqrt{\langle \boldsymbol{\Theta}, \boldsymbol{\Theta} \rangle}$. This is equivalent to identifying $\mathcal{M}_p$ with the Euclidean space $\mathbb{R}^{p(p+1)/2}$. For $\boldsymbol{\Theta} \in \mathcal{M}_p$, we also define that $\|\boldsymbol{\Theta}\|_1 := \sum_{j \le k} |\theta_{jk}|$.

Building upon the aforementioned notations, we can propose the estimates for the change points in the time-varying Markov Random Field and their corresponding graph structure estimates. Let $L_T(I_j(\boldsymbol{\alpha}), \boldsymbol{\Theta})$ be the loss function for the j-th interval of the candidate partition α, which is defined as:

$$L_T(I_j(\boldsymbol{\alpha}), \boldsymbol{\Theta}) = \frac{1}{T} \Phi_{(\alpha_{j-1}, \alpha_j]}(\boldsymbol{\Theta}) = \frac{1}{T} \sum_{i=\alpha_{j-1}T+1}^{\alpha_j T} \phi(\boldsymbol{\Theta}, \boldsymbol{X}^{(i)}). \tag{2.10}$$

Then, given hyper-parameters $\lambda > 0$, $\gamma > 0$ and $\delta > 0$, the joint estimator of chang points and the associated graph structures of the time-varying Markov Random Fields is as follows

$$\widehat{\alpha} = \arg\min_{k} \arg\min_{\alpha : l(\boldsymbol{\alpha}) = k} \Big\{ \sum_{j=1}^{k} L_T(I_j(\boldsymbol{\alpha}), \widehat{\boldsymbol{\Omega}}^{(j)}) + \gamma k \Big\}, \tag{2.11}$$

where

$$\widehat{\boldsymbol{\Omega}}^{(j)} = \arg\min_{\boldsymbol{\Theta}} \Big\{ L_T(I_j(\boldsymbol{\alpha}), \boldsymbol{\Theta}) + \lambda \sqrt{r_j(\boldsymbol{\alpha})} \|\boldsymbol{\Theta}\|_1 \Big\}, \quad j = 1, \ldots, l(\boldsymbol{\alpha}), \tag{2.12}$$

Hence, we employ the Pseudo-Likelihood method to derive joint estimates for the change points and graph structures in the time-varying Markov Random Fields, which is essentially a series of optimization problems. The minimization in (2.11) is carried out over the set of all vectors $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_k)$ satisfying $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_k = 1$ and $r(\boldsymbol{\alpha}) \geq \delta$. The role of δ is to ensure that within each segment (between two continuous candidates of change points), there are an adequate number of samples, providing a reasonable guarantee of accuracy for the estimation of graph structures. Additionally, we impose an upper bound on k determined by the minimal spacing $r(\boldsymbol{\alpha}) \geq \delta$, implying that $k \leq 1/\delta$. We will delve into the computation of $\widehat{\boldsymbol{\Omega}}^{(j)}$ in detail in Section 3. First, assuming that we have solved the optimization problem (2.12), we focus on tackling the optimization problem in (2.11) and introduce two algorithms to find the best candidate for the change point vector $\widehat{\boldsymbol{\alpha}}$.

2.3.1 Dynamic programming

We present a comprehensive approach based on the dynamic programming algorithm, which is a popular technique for multiple change point detection (Friedrich et al., 2008; Killick et al., 2012; Leonardi and Bühlmann, 2016). We adopt it for the time-varying Markov random fields. The dynamic programming algorithm is renowned for its exceptional accuracy, as it takes into account the global solution of (2.11). It computes the optimum in (2.11) and the estimates in (2.12) with a computational cost of $\mathcal{O}(T^2 Penalty(T))$ operations, where $Penalty(T)$ represents the time cost of solving the estimation of $\boldsymbol{\Omega}$ with L_1 penalty in (2.12), for the samples of size T.

For any given $v \in (0, 1]$, we define $F_k(v)$ as the minimum value of the function in (2.11) when considering only the sample $\boldsymbol{X}^{(0,v]}(v \leq 1)$ and vectors α of size $l(\boldsymbol{\alpha}) = k$. To be more specific, $F_k(v)$ has the following form

$$F_k(v) = \min_{\boldsymbol{\alpha}: l(\boldsymbol{\alpha})=k} \left\{ \sum_{j=1}^{k} L_T(I_j(v\boldsymbol{\alpha}), \widehat{\boldsymbol{\Omega}}^{I_j(v\boldsymbol{\alpha})}) + \gamma k \right\}. \tag{2.13}$$

Note that by its definition, $F_k(v)$ is the minimum loss of dividing the samples $\boldsymbol{X}^{(0,v]}(v \leq 1)$

into k segments. Moreover, from the above definition, we know that $I_j(v\boldsymbol{\alpha})$ is the j-th interval $(v\alpha_{j-1}, v\alpha_j]$, therefore we denote $\boldsymbol{\Omega}^{I_j(v\boldsymbol{\alpha})}$ as the graph structure for the observations $\boldsymbol{X}^{I_j(v\boldsymbol{\alpha})}$ assuming that this interval are static Markov random fields and $\widehat{\boldsymbol{\Omega}}^{I_j(v\boldsymbol{\alpha})}$ is the corresponding estimates. For any $v \in V_T = \{i/T : i = 1, 2, \ldots, T\}$, define

$$F_1(v) = \begin{cases} +\infty, & v < \delta \\ L_T\big((0, v], \widehat{\boldsymbol{\Omega}}^{(0,v]}\big) + \gamma, & v \geq \delta \end{cases} \tag{2.14}$$

Then, we can establish the dynamic programming recursion as follows:

$$F_k(v) = \min_{u \in V_T, u < v} \left\{ F_{k-1}(u) + L_T\big((u, v], \widehat{\boldsymbol{\Omega}}^{(u,v]}\big) + \gamma \right\}, \quad k = 2, \cdots, k_{max}. \tag{2.15}$$

where k_{max} serves as an upper limit on the value of k (in our constraint, $k_{max} = 1/\delta$). Leveraging equations (2.14) and (2.15), we can compute $\{F_1(v), v \in V_T\}, \{F_2(v), v \in V_T\}, \ldots,$ and $\{F_{k_{max}}(v), v \in V_T\}$. Consequently, we can determine the optimal value of k through the equation

$$\widehat{k} = \operatorname*{arg\,min}_{k=1,\ldots,k_{max}} \left\{ F_k(1) \right\}, \tag{2.16}$$

and the corresponding estimate of change point vector $\widehat{\boldsymbol{\alpha}} = (0, \widehat{\alpha}_1, \cdots, \widehat{\alpha}_{\widehat{k}-1}, 1)$ is given by

$$\widehat{\alpha}_{j-1} = \operatorname*{arg\,min}_{u \in V_T, u < \widehat{\alpha}_j} \left\{ F_{j-1}(u) + L_T\big((u, \widehat{\alpha}_j], \widehat{\boldsymbol{\Omega}}^{(u,\widehat{\alpha}_j]}\big) + \gamma \right\}, \quad j = \widehat{k}, \ldots, 1. \tag{2.17}$$

The steps of the dynamic programming algorithm can be outlined as follows.

Algorithm 1 Dynamic Programming for high dimensional Markov random field models

1: Given the value of k_{max}, for $k = 1, \cdots, k_{max}$, compute $F_k(1)$ for $k = 1, \ldots, k_{max}$ based on (2.14) and (2.15);

2: Obtain the estimate of the number of change points $\widehat{k}$ by (2.16);

3: Obtain the estimate of the change point locations $\widehat{\boldsymbol{\alpha}}$ by (2.17).

2.3.2 Binary segmentation

Although the dynamic programming algorithm can realize high precision solutions, its computational cost is pretty high. We introduce a much more efficient approach — binary

segmentation algorithm, to solve the same problem. Binary segmentation is a widely used technique that can be adapted to address a variety of problems related to multiple change point detection. For instance, Leonardi and Bühlmann (2016) introduced the principles of binary segmentation, specifically designed for detecting change points in high-dimensional regression settings. In this article we want to utilize the method in our data of Markov random fields. It incurs a computational cost of $\mathcal{O}(T\log(T)Penalty(T))$, where $Penalty(T)$ represents the expense of computing the estimator of $\mathbf{\Omega}$ with L_1 penalties, as illustrated in (2.12), for the samples of size T.

For $u, v \in V_T = \{i/T : i = 1, 2, \ldots, T\}$ we denote $\mathbf{\Omega}^{(u,v]}$ as the graph structure for the observations $\boldsymbol{X}^{(u,v]}$ assuming that this interval are static Markov random fields. We define

$$H(u,v) = \begin{cases} L_T((u,v], \widehat{\mathbf{\Omega}}^{(u,v]}) + \gamma, & (v-u)T \geq 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.18}$$

and

$$h(u,v) = \underset{s\in\{u\}\cup[u+\delta, v-\delta]}{\arg\min} \Big\{ H(u,s) + H(s,v) \Big\}. \tag{2.19}$$

The core concept of binary segmentation revolves around identifying the optimal single change point within the interval $(0, 1]$, with the constraint of $h(0, 1) \neq 0$. Next, we continually search for new single change points on both sides of the segmentation generated by the initial single change point, akin to the generation of a binary tree structure. The decision to create sub-segments is based on a certain metric, and this process continues until no new sub-segments are generated (no new change points emerge). It is evident that this metric must incorporate a regularization penalty term to prevent overfitting.

We can describe this algorithm using a binary tree structure denoted as $\mathcal{T}$. Adopting the above notation $V_T = \{i/T : i = 1, 2, \cdots, T\}$, we define $(u, v] \in V_T^2$ subject to the constraint $(v - u)T \geq 1$ and use $(u, v]$ to be the label of a node in the binary tree. The algorithm's steps are outlined below.

Algorithm 2 Binary Segmentation for high dimensional Markov random field models

1: Initialize $\mathcal{T}$ to the tree with a single root node labeled by $(0, 1]$;

2: For each terminal node $(u, v]$ in $\mathcal{T}$ compute $s = h(u, v)$. If $s > u$ add to $\mathcal{T}$ the new nodes $(u, s]$ and $(s, v]$ as descendants of the node $(u, v]$;

3: Repeat 2. until no more nodes can be added to $\mathcal{T}$.

The collection of terminal nodes within $\mathcal{T}$, represented as $\mathcal{T}^0$, will yield the estimate for change point vector $\widehat{\alpha}^{bs}$ by selecting the boundaries within these intervals; specifically,

$$\widehat{\boldsymbol{\alpha}}^{bs} = \bigcup_{(u,v]\in\mathcal{T}^0} \{u, v\}. \tag{2.20}$$

3 Improvement on estimation for graph structure

In Section 2, we introduced the joint estimation of change points and graph structures for the time-varying Markov random fields, as outlined in (2.11). Assuming we have obtained reasonably accurate estimates for $\{\Omega^{(j)}, j = 1, \cdots, l(\boldsymbol{\alpha})\}$, utilizing the two aforementioned optimization algorithms can yield the optimal number of change points and their positions. Therefore, in this chapter, our primary focus is on obtaining estimates for a series of graph structures $\{\Omega^{(j)}, j = 1, \cdots, l(\boldsymbol{\alpha})\}$. Höfling and Tibshirani (2009) proposed an algorithm based on Pseudo-Likelihood to compute the graph structures with L_1 penalty, as defined in (2.12). This algorithm is available in the `BMN` package in R.

However, when applied to change point estimation problems, `BMN` also has its limitations. Firstly, it requires a significant number of samples to estimate the graph structures accurately (assuming these samples are within a certain segment, i.e., share the same graph structure). Therefore, when we aim to estimate within shorter intervals, corresponding to as few samples as possible, the precision of graph structure estimation dramatically decreases. Secondly, in the realm of optimization problems, `BMN` incurs a high computational cost. The change point estimation problem, particularly in the case of multiple change

points, demands the development of more efficient and faster optimization algorithms suitable for the Pseudo-Likelihood (PL) method. As mentioned above, the objective function of pseudo-likelihood shares a structural similarity with the loss function of logistic regression. Geng et al. (2017) has come up with a new method to achieve the transformation by means of similarity and symmetry of this two functions. Nevertheless, this transformation is developed for static Markov Random Fields, which does not take into account the temporal scale order-preserving property, making the chronologically arranged samples disordered after the transformation. Based on this, we have modified and improved the algorithm for graph structure estimation, by transforming the Markov Random Field observations into the logistic regression observations in the same chronological order, and then plug this new algorithm into change point detection framework.

3.1 Pseudo-likelihood objective function

Let's begin by examining the objective function of the PL method. We consider a consecutive time interval from the whole observations, assuming that those Markov random fields are static and there are N samples. Obviously these N samples follow the same graph structure. We denote these N samples $\{\boldsymbol{X}_i^*\}_{i=1}^N$ as a matrix $\mathbf{X}^*$ and it has the following form,

$$\mathbf{X}^* = \begin{bmatrix} (\boldsymbol{X}_1^*)^\top \\ (\boldsymbol{X}_2^*)^\top \\ \vdots \\ (\boldsymbol{X}_N^*)^\top \end{bmatrix} = \begin{bmatrix} x_{11}^* & x_{12}^* & \dots & x_{1p}^* \\ x_{21}^* & x_{22}^* & \dots & x_{2p}^* \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1}^* & x_{N2}^* & \dots & x_{Np}^* \end{bmatrix} = \left[x_{ij}^*\right]_{N\times p}, \tag{3.1}$$

in which each row corresponds to a sample at a time point, representing a static Markov Random Field. The associated weight symmetric matrix $\mathbf{\Theta}^* \in \mathbb{R}^{p\times p}$ is given as

$$\mathbf{\Theta}^* = \begin{bmatrix} \theta^*_{11} & \theta^*_{12} & \dots & \theta^*_{1p} \\ \theta^*_{21} & \theta^*_{22} & \dots & \theta^*_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \theta^*_{p1} & \theta^*_{p2} & \dots & \theta^*_{pp} \end{bmatrix} = \left[\theta^*_{ij}\right]_{p\times p}. \tag{3.2}$$

According to Definition 2.2, we have the negative log-pseudo-likelihood function of these N samples as follows,

$$\phi(\mathbf{\Theta}^*, \mathbf{X}^*) = -\sum_{n=1}^{N}\sum_{s=1}^{p}\left[x^*_{ns}\left(\theta^*_{ss} + \sum_{t\neq s} x^*_{nt}\theta^*_{st}\right) - \log\left(1 + \exp(\theta^*_{ss} + \sum_{t\neq s} x^*_{nt}\theta^*_{st})\right)\right]. \tag{3.3}$$

If we add L_1 penalty in (2.12) to the function, the objective function becomes

$$\mathcal{L}(\mathbf{\Theta}^*, \mathbf{X}^*) = \phi(\mathbf{\Theta}^*, \mathbf{X}^*) + \lambda T\sqrt{r_j(\boldsymbol{\alpha})}\|\mathbf{\Theta}^*\|_1 = \phi(\mathbf{\Theta}^*, \boldsymbol{X}^*) + \lambda'\sum_{s\leq t}|\theta^*_{st}|.$$

Here, we replace the original $\lambda T\sqrt{r_j(\boldsymbol{\alpha})}$ with λ' and get the objective function of the PL method.

3.2 Conversion from Pseudo-likelihood to logistic regression

Now, we delve into the relationship between the objective function of Pseudo-Likelihood and the loss function of logistic regression. We will demonstrate that by constructing appropriate $\widetilde{\mathbf{\Theta}}$, $\widetilde{\mathbf{X}}$, and response variable $\widetilde{\boldsymbol{y}}$, we can transform the PL problem with L_1 penalty into a logistic regression (LR) problem with L_1 penalty, involving regression parameters $\widetilde{\mathbf{\Theta}}$, covariates $\widetilde{\mathbf{X}}$, and response variable $\widetilde{\boldsymbol{y}}$.

We begin by constructing $\widetilde{\mathbf{\Theta}}$. Since in logistic regression, regression parameters form a vector rather than a matrix, we need to transform the graph structure $\mathbf{\Theta}^*$ into a vector. Due to its symmetry, we can stack its upper triangular elements in a column-wise manner and then sequentially append the diagonal elements to the end of the vector to form $\widetilde{\mathbf{\Theta}}$ as follows:

$$\widetilde{\Theta} = \Big(\overbrace{\theta^*_{12},\ \theta^*_{13},\ \theta^*_{23},\ \theta^*_{14},\ \ldots,\ \theta^*_{(p-2)p},\ \theta^*_{(p-1)p}}^{\text{upper triangular elements}},\ \overbrace{\theta^*_{11},\ \ldots,\ \theta^*_{pp}}^{\text{diagonal elements}}\Big) \in \mathbb{R}^{p(p+1)/2}, \quad (3.4)$$

where for any $(a,b) \in \{(a,b)|a \neq b, (a,b) \in V \times V\}$, θ^*_{ab} is converted into the j-th element in $\widetilde{\mathbf{\Theta}}$ in which we can define this mapping as $f : V \times V \longrightarrow \mathbb{R}$ and specify as follows

$$j = f(a,b) = \min(a,b) + \frac{(\max(a,b)-1)(\max(a,b)-2)}{2}, j = 1, \ldots, p(p-1)/2. \quad (3.5)$$

It is noteworthy that $f(a,b) = f(b,a)$. Let us construct the matrix $\widetilde{\mathbf{X}} = [\widetilde{x}_{ij}]_{Np\times(m+p)}$, in which $m = \frac{p(p-1)}{2}$ represents the number of upper triangular elements of graph structure Θ^*. Based on this, we define

$$\widetilde{x}_{ij} = \begin{cases} x^*_{nb} & \exists\,(a,b) \text{ s.t. } j = f(a,b) \\ \mathbb{I}(a = j-m) & m+1 \leqslant j \leqslant m+p \\ 0 & \text{otherwise} \end{cases}, \quad (3.6)$$

where $i = p(n-1)+a$, $n \in \{1,2,\ldots,N\}, a \in \{1,2,\cdots,p\}, b \in \{1,2,\cdots,p\}$ and $\mathbb{I}(\cdot)$ is the indicator function.

It may seem to be difficult to understand. Let us look at this conversion in more detail. We set $\widetilde{\mathbf{X}}$ with the dimension $Np \times (m+p)$, which can be regarded as N submatrices of size $p \times (m+p)$ stacked vertically. Considering the (i,j) element $\widetilde{x}_{ij}$, in the row index formula $i = p(n-1)+a, n \in \{1,2,\cdots,N\}, a \in \{1,2,\cdots,p\}$, for a specific value of a, we can find that N row indices are generated – $\{a, p+a, 2p+a, \cdots, p(N-1)+a\}$. These indices correspond to the same row in different submatrices of size $p \times (m+p)$, meaning that the elements in the same row of each submatrix will have the same value if they are in the same column. Specifically, we can write the pseudocode for the transformation as in Algorithm 3.

Algorithm 3 Construct the covariates of LR from the PL

Set $\widetilde{\mathbf{X}} \in \mathbb{R}^{Np\times(m+p)}$ to be a matrix whose items are all 0

for $j = 1, 2, \cdots, m$ with $m = p(p-1)/2$ **do**

Find $(a, b) \in \{(s, t)|s \neq t, (s, t) \in V \times V\}$ such that $j = f(a, b)$

for $n = 1, 2, \cdots, N$ **do**

Set $i = p(n-1) + a$, construct $\widetilde{x}_{ij} = x^*_{nb}$

end for

end for

for $j = m+1, \cdots, m+p$ **do**

for $a = 1, 2, \cdots, p$ **do**

if $a = j - m$ **then**

for $n = 1, 2, \cdots, N$ **do**

Set $i = p(n-1) + a$, $\widetilde{x}_{ij} = 1$

end for

end if

end for

end for

We have explained our conversion rules from $\mathbf{X}^*$ to $\widetilde{\mathbf{X}}$. Ultimately, we define the response $\widetilde{\boldsymbol{y}}$ as

$$\widetilde{\boldsymbol{y}} = \left(x^*_{11}, x^*_{12}, \ldots, x^*_{1p}, \ldots, x^*_{N1}, x^*_{N2}, \ldots, x^*_{Np}\right)^\top. \tag{3.7}$$

If we denote $(\widetilde{\mathbf{X}})_k$ as the transposition of k-th row vector of $\widetilde{\mathbf{X}}$

$$(\widetilde{\mathbf{X}})_k = (\widetilde{\mathbf{X}}_{k1}, \widetilde{\mathbf{X}}_{k2}, \ldots, \widetilde{\mathbf{X}}_{k(m+p)})^\top, \tag{3.8}$$

we can prove that

$$\sum_{n=1}^{N}\sum_{s=1}^{p}\left[x_{ns}^{*}\left(\theta_{ss}^{*}+\sum_{t\neq s}x_{nt}^{*}\theta_{st}^{*}\right)\right]=\widetilde{\boldsymbol{y}}^{\top}\widetilde{\mathbf{X}}\widetilde{\boldsymbol{\Theta}} \tag{3.9}$$

$$\sum_{n=1}^{N}\sum_{k=1}^{p}\left[\log\left(1+\exp(\theta_{kk}^{*}+\sum_{t\neq k}x_{nt}^{*}\theta_{kt}^{*})\right)\right]=\sum_{k=1}^{Np}\log\left(1+\exp((\widetilde{\mathbf{X}})_{k}^{\top}\widetilde{\boldsymbol{\Theta}})\right). \tag{3.10}$$

With the design matrix $\widetilde{\mathbf{X}}$, parameters $\widetilde{\boldsymbol{\Theta}}$, and response $\widetilde{\boldsymbol{y}}$, we can rewrite the original objective function of Markov random fields in (3.3) as:

$$\phi(\boldsymbol{\Theta}^{*},\mathbf{X}^{*})=-\left[\widetilde{\boldsymbol{y}}^{\top}\widetilde{\mathbf{X}}\widetilde{\boldsymbol{\Theta}}-\sum_{k=1}^{Np}\log\left(1+\exp((\widetilde{\mathbf{X}})_{k}^{\top}\widetilde{\boldsymbol{\Theta}})\right)\right]. \tag{3.11}$$

If we add L_1 penalty with hyper-parameter λ', we can rewrite it as

$$\mathcal{L}(\boldsymbol{\Theta}^{*},\mathbf{X}^{*})=-\left[\widetilde{\boldsymbol{y}}^{\top}\widetilde{\mathbf{X}}\widetilde{\boldsymbol{\Theta}}-\sum_{k=1}^{Np}\log\left(1+\exp((\widetilde{\mathbf{X}})_{k}^{\top}\widetilde{\boldsymbol{\Theta}})\right)\right]+\lambda'\|\widetilde{\boldsymbol{\Theta}}\|_{1}, \tag{3.12}$$

which is exactly the loss function for a L_1 penalized logistic regression problem consisting of Np samples with the design matrix $\widetilde{\mathbf{X}}$, response $\widetilde{\boldsymbol{y}}$ and the parameters $\widetilde{\boldsymbol{\Theta}}$. It is noticed that minimizing the L_1 penalty negative log pseudo-likelihood function is equal to minimizing the loss function of a L_1 penalized logistic regression. Therefore, we have achieved the equivalent conversion of the problem. In the case of logistic regression, more advanced optimization packages, such as `glmnet`, can be used to obtain solutions more efficiently and with higher accuracy.

3.3 The meaning of equivalent conversion

In summary, we propose an innovative equivalence transformation method which preserves the time order, designed for estimating static Markov random fields. In this section we will integrate the graph structure estimation of static Markov random fields with the optimal change-point search algorithm discussed earlier, introducing several strategies for change-point detection. By comparing these strategies, we will elaborate the significance of this time-order-preserving equivalence transformation step by step.

The optimization problem of change point detection is characterized by the objective functions (2.11) and (2.12). The whole optimization can be divided into two parts —— change-point search and graph structure estimation for each piecewise static Markov random fields. There are two algorithms to carry out the optimal change-point search, which are dynamic programming and binary segmentation. The two algorithms give different approaches for searching potential change points. For each piecewise interval between two consecutive change points, both of two algorithms require estimation of the corresponding graph structures. We shall take binary segmentation algorithm as an example, integrating it with several methods of static Markov random field estimation, in order to propose a series of change-point detection strategies and evaluate their own performances.

At first for the estimation of static Markov random fields, we know that `BMN` package in R is developed to solve the problem by (Höfling and Tibshirani, 2009). The strategy integrating BMN into binary segmentation algorithm is as follows.

Stratgey 1: integrate BMN with binary segmentation

Step 1: Assume there are T samples. On the full sample interval $[1, T]$, binary segmentation can split the original interval into a series of left and right subintervals.

Step 2: For each pair of subintervals (left and right), BMN is used to estimate the corresponding graph structure, thereby calculating the loss of this splitting way.

Step 3: From the series of splitting ways, the one with the smallest loss is selected, dividing the initial interval $[1, T]$ into a left-subinterval and a right-subinterval.

Step 4: On the corresponding left-subinterval and right-subinterval, steps 1-3 are repeated, until no further splitting is allowed (there is the regularity condition in the previous text).

We consider the computational cost in strategy 1, which is primarily determined by

the number of times the change-point candidate set is traversed and the binary splits. From previous section, we know that the computational cost of binary segmentation is $\mathcal{O}(T\log(T)Penalty(T))$, where $Penalty(T)$ represents the time cost of estimating the graphical structures with L_1 penalty in (2.12) for the samples of size T. In this BMN case, it is exactly the BMN method time cost to solve the estimation of static Markov random fields for T samples, which is denoted as $BMN(T)$. Therefore, the time cost in strategy 1 is $\mathcal{O}(T\log(T)BMN(T))$.

However, with the development of optimization methods, `BMN` has gradually revealed several drawbacks. It demands a large number of samples in order to accurately estimate the graph structures. When estimating over shorter intervals with fewer samples, the precision significantly drops. Additionally, `BMN` also incurs high computational costs, which means $BMN(T)$ cost a lot of time.

To reduce the time cost, we seek alternative optimization methods for estimating the graph structure of static Markov random fields. Given the symmetry between Markov random fields and logistic regression, previous work has shown that Markov random fields can be converted into logistic regression (Geng et al., 2017). However, in the context of time-varying Markov random fields, this conversion does not guarantee the preservation of time order. For example, suppose we have performed an equivalence transformation on the full sample interval $[1, T]$ to convert it into a logistic regression sample, denoted as $LR[1, T]$. When we want to estimate the graph structure on a subinterval $[T_0, T_1]$ with $1 < T_0 < T_1 < T$, we cannot directly extract the corresponding subset from $LR[1, T]$, as the sample order in $LR[1, T]$ is scrambled and no longer aligns with the order in [1,T]. Therefore, we need to perform the equivalence transformation on $[T_0, T_1]$ to obtain $LR[T_0, T_1]$. That is to say, $LR[T_0, T_1] \not\subset LR[1, T]$. We call this equivalence transformation as PL-LG without preserving time-order. Therefore it comes to our second strategy of change point detection. The first step is omitted here due to its similarity.

Stratgey 2: integrate PL-LG without preserving time-order with binary segmentation

Step 1: Same as Step 1 in Strategy 1.

Step 2: For each pair of subintervals (left and right), the equivalence transformation is carried out to get the corresponding logistic samples. We use `glmnet` to estimate the logistic regression as well as the corresponding graph structure, thereby calculating the loss of this splitting way.

Step 3: From the series of splitting ways, the one with the smallest loss is selected, dividing the initial interval $[1, T]$ into a left-subinterval and a right-subinterval.

Step 4: On the corresponding left-subinterval and right-subinterval, steps 1-3 are repeated, until no further splitting is allowed. It should be noticed in each splitting operation, the equivalence transformation should be done to the corresponding splitting interval.

Let's consider the time complexity of strategy 2. The time complexity arising from splitting and estimation is $\mathcal{O}(T \log(T) Penalty(T))$, where in transformation case $Penalty(T)$ is exactly the time cost of estimating the converted logistic regression from Markov random fields of size T. We adopt `glmnet` in R to solve logistic regression estimation, which costs less time and only needs few samples to estimate accurately. However, it requires an equivalence transformation at each split, which adds an additional time cost of $\mathcal{O}(T \log(T))$ equivalence transformations. Actually this is a significant time expense.

After introducing strategies 1 and 2, it comes to our innovative method — convert Markov random fields (MRFs) into logistic regression with the time-order preserved. From the previous transformation algorithm 3, we know that T MRF observations are actually converted into $T * p$ logistic regression samples, with p be the dimension of MRF, meaning that each MRF observation is transformed into p logistic regression samples. Most importantly, the time order is preserved. If we use $[1, T]$ to represent all MRF samples,

the transformed logistic regression samples are denoted as $LR[1, T*p]$. When performing equivalence transformation on the MRF within a subinterval $[T_0, T_1]$ with $1 < T_0 < T_1 < T$, we can simply select the subset $LR[(T_0-1)*p+1, T_1*p] \subset LR[1, T*p]$, without needing to perform a new equivalence transformation. The corresponding strategy is as follows.

Stratgey 3: integrate PL-LG with preserving time-order with binary segmentation

Step 1: Assume there are T samples. On the full sample interval $[1, T]$, we firstly carry out the equivalent transformation to convert them into logistic regression samples $LR[1, T*p]$. Then binary segmentation can split the original interval into a series of left and right subintervals.

Step 2: For each pair of subintervals (left and right), we refer to the appropriate subset of $LR[1, T*p]$ to get the corresponding logistic regression samples. We use `glmnet` to estimate the logistic regression as well as the corresponding graph structure, thereby calculating the loss of this splitting way.

Step 3: From the series of splitting ways, the one with the smallest loss is selected, dividing the initial interval $[1, T]$ into a left-subinterval and a right-subinterval.

Step 4: On the corresponding left-subinterval and right-subinterval, steps 2-3 are repeated, until no further splitting is allowed. It should be noticed in each splitting operation, there is no need for the equivalence transformation since we can just select the appropriate subset of $LR[1, T*p]$ to get the converted logistic regression samples.

In strategy 3, the time complexity arising from splitting and estimation is $\mathcal{O}(T\log(T)Penalty(T))$, where $Penalty(T)$ is exactly the time cost of estimating the converted logistic regression from Markov random fields of size T. Strategy 3 requires only one equivalence transformation, instead of $\mathcal{O}(T\log(T))$ transformation in strategy 2. It fully leverages the time

and accuracy advantages of equivalence transformation for estimating the static Markov random field. Since our method uses an order-preserving transformation, only a single equivalence transformation is required for the entire MRF samples before starting any change-point search algorithms. We can simply access the appropriate logistic regression subset to estimate the Markov random field for the corresponding interval.

In summary, we have demonstrated that parameter estimation for a static Markov random field (employing the PL method) can be transformed into parameter estimation for logistic regression utilizing their consistency of loss function. Distinct from previous transformation, our developed transformation preserves the time-order for samples. Three strategies are constructed to illustrate why our transformation method is prominent in the whole change point detection process, owing to its more refined grid search, better estimation accuracy and less time cost.

More importantly, we believe that this equivalence transformation goes beyond a mere optimization conversion. The time-order preserved property guarantees that we can extend a parameter estimation problem to a multiple change-point detection problem in both Markov random fields and logistic regression. We believe it warrants further in-depth research and exploration beyond the scope of this article.

Additionally, it is noteworthy that these three strategies take binary segmentation algorithm as an instance to analyze the time complexity integrated with each static MRF estimation methods. The analysis of dynamic programming algorithms can be analogous. In view of these strategies, we have obtained a robust estimation of the graph structure. When combined with the dynamic programming and binary segmentation algorithms discussed earlier, we can then compute the optimal number and positions of latent change points, represented as the change point vector shown in (2.11).

4 Numerical experiment

In the preceding sections, we established the model for the time-varying Markov Random Fields along with its learning strategies. Furthermore, we introduced an innovative transformation of the objective function that effectively converts the Markov Random Field into a logistic regression, thereby improving both solution efficiency and accuracy. In this section, we will conduct numerical experiments to assess the effectiveness of our proposed learning strategies and to validate the comparative performance of various algorithms for estimating the graph structure.

4.1 Hyper-parameter settings

Based on the joint estimation functions for change points and the graph structure in the time-varying Markov random field, as expressed in Equations (2.11) and (2.12), we introduce hyper-parameters λ and γ. Clearly, γ is used to control the number of change points (i.e., the size of segments), while λ serves as a regularization parameter to prevent overfitting when estimating the graph structure within each segment (an assumption aligned with the widely recognized sparse graph hypothesis). In our numerical experiments, we choose these hyper-parameters based on the Bayesian Information Criterion (BIC) criterion according to (Roy et al., 2017). Recalling that our estimated change point vector is denoted as $\widehat{\alpha} = (\alpha_0, \alpha_1, \alpha_2, \cdots)$ $(\alpha_0 = 0)$, we denote $l(\widehat{\alpha})$ as the number of segments derived from this change point estimator and denote its corresponding graph structures as $\{\widehat{\Omega}^{(j)}, j = 1, \ldots, l(\widehat{\alpha})\}$. We can formulate the BIC criterion as follows:

$$\mathrm{BIC}(\widehat{\alpha}, \widehat{\Omega}^{(1)}, \cdots, \widehat{\Omega}^{(l(\widehat{\alpha}))}) \triangleq \sum_{j=1}^{l(\widehat{\alpha})} \left(2L_T(I_j(\widehat{\alpha}), \widehat{\Omega}^{(j)}) + \log(T\alpha_j - T\alpha_{j-1}) \|\widehat{\Omega}^{(j)}\|_0 \right) \tag{4.1}$$

where $T\alpha_j - T\alpha_{j-1}$ represents the number of observations within the j_{th} segment $(\alpha_{j-1}, \alpha_j]$. If we denote $\widehat{\Omega}^{(j)}$ as

$$\widehat{\Omega}^{(j)} = \left[\widehat{\omega}_{ab}^{(j)} \right]_{p \times p}, \quad j = 1, \cdots, l(\widehat{\alpha})$$

we have

$$\|\widehat{\Omega}^{(j)}\|_0 \coloneqq \sum_{a \le b \le p} \mathbb{I}\{|\widehat{\omega}_{ab}^{(j)}| > 0\}.$$

We also prepare a second method for tuning parameters —- cross-validation (CV), which presupposes that multiple samples are generated at each time point $i = 1, \cdots, T$. This allows the time series to be divided into a learning phase and a testing phase. In this framework, we select the hyper-parameters which maximize the AUC, that is, the area under the ROC curve corresponding to the classification score.

For both BIC and CV model selection techniques, we employ a grid search approach to select hyper-parameters which minimize the BIC criterion or maximize the AUC criterion, thereby obtaining the final estimates for change points and graph structures.

4.2 Measuring metrics

We employ various metrics to assess the quality of change point and graph structure estimates. First, we consider a commonly used measure for change point estimation, the Hausdorff metric (Truong et al., 2019) Assuming that the true set of change points is denoted as D, and the estimated set of change points is $\widehat{D}$, we have

$$h(\mathcal{D}, \widehat{\mathcal{D}}) \triangleq \frac{1}{T} \max \left\{ \max_{t \in \mathcal{D}} \min_{\widehat{t} \in \widehat{\mathcal{D}}} |t - \widehat{t}|, \ \max_{\widehat{t} \in \widehat{\mathcal{D}}} \min_{t \in \mathcal{D}} |t - \widehat{t}| \right\}.$$

We refer to it as the *h-score*, where it's evident that a smaller *h-score* indicates more accurate change point estimation.

We consider the estimation of the graph structure. Since the graph structure represents the pairwise conditional dependency relationships between nodes, which are the edges in the graph structure, we need appropriate criteria to measure the quality of learning these edges within the graph structure. At each time point in the time series, there exists a set of true edges and a set of edges generated based on the estimated graph structure. As defined earlier, let's denote the time-varying Markov random field as $\{\boldsymbol{X}^{(i)}, 1 \le i \le T\}$,

with a total of T samples. We denote the set of true edges as $\{E_i\}_{i=1}^T$ and the set of edges generated based on the estimated graph structure as $\{\widehat{E}_i\}_{i=1}^T$, i.e.

$$\widehat{E}_i = \left\{(a,b) : |\widehat{\omega}_{ab}^{(i)}| > 0\right\}, i = 1, \cdots, T.$$

Here we have some comments. Our proposed method ensures the symmetry of the graph structure in the estimation process. However, some other methods such as neighborhood selection strategy may attain to an asymmetric graph structure, which will be mentioned later in the baseline method part.

Next, we introduce the commonly used metric in classification, the F_1-*score*, to assess the quality of the learned graph structure:

$$F_1 = \frac{2 \times precision \times recall}{precision + recall},$$

which comprises two terms as follows,

$$precision = \frac{1}{T}\sum_{i=1}^{\top}\sum_{a<b}\frac{\mathbb{I}\{(a,b) \in \widehat{E}_i \wedge (a,b) \in E_i\}}{\mathbb{I}\{(a,b) \in \widehat{E}_i\}},$$
$$recall = \frac{1}{T}\sum_{i=1}^{\top}\sum_{a<b}\frac{\mathbb{I}\{(a,b) \in \widehat{E}_i \wedge (a,b) \in E_i\}}{\mathbb{I}\{(a,b) \in E_i\}}.$$

4.3 Baseline methods

As mentioned in Section 1, there are many static network estimation methods in the past. However, there are few works in the change-point detection framework. The closest work we can compare with ours is TVI-FL method proposed by (Le Bars et al., 2020). TVI-FL is a method using group fused lasso to recover change points as well as the graph structures. The main difference is that TVI-FL assumes that the graph structure of the consecutive segments only changes some parts of the edges at the change point, leaving the other edges unchanged, while our method impose nothing on the network structure changes in the time period before and after the change point. When involving static estimation of the graph structure, TVI-FL adopts the neighborhood selection strategy, which means some

asymmetric results about the graph structure, thus requiring some adjustments for our assessment metrics

$$\widehat{E}_i = \left\{(a,b) : \max\left(|\widehat{\omega}_{ab}^{(i)}|, |\widehat{\omega}_{ba}^{(i)}|\right) > 0\right\}, \quad i = 1, \cdots, T.$$

From the perspective of computational cost, TVI-FL method needs to handle the total loss with the L_1 penalty and fused lasso at the same time, while our method proposes optimization algorithms to consider separately the sparsity of the graph and the penalty of the number of change points. This will lead to significant complexity gap. We shall compare the performance of baseline methods and our methods in the following numerical experiment.

4.4 Synthetic data

We set the length of the time series to be 600 time points, with $|D| = 3$, meaning there are three change points, occurring at the 101-st, 251-st and 501-st time points, resulting in four segments (with segment lengths of 100, 150, 250, and 100 respectively).

Next, We consider how to generate the corresponding graph structures of each segment. We adopt the idea of regular graph. The graph structures for those segments are generated to be independent random d-regular graph of p nodes, where p is the number of nodes in the network and d is the degree of the all nodes. These edges exclude the self-connected edges, which means the diagonal terms of the graph structure are set to 0. For those existing edges generated by a regular graph, their specific weights are randomly sampled from a uniform distribution in the range $[-1, -0.5] \cup [0.5, 1]$. The values of p and d are parameters that can be controlled.

Once we have the graph structures, we obtain samples from the Markov Random Field using *Gibbs Sampling*. We set a burn-in length of 2000 samples, and to avoid dependence between samples, we extract one sample every 20 samples.

4.5 Numerical results

Before turning to the simulation process and results, we firstly do a brief review on our methods. We have proposed a joint estimator for the change points and the corresponding graph structures in Equation (2.11). For the estimation of change points, which is essentially an optimization problem, there are two algorithms to tackle with —- dynamic programming and binary segmentation. These two algorithms can achieve similar performances especially when the data shows a strong signal following the underlying graph structures. Since binary segmentation is much faster than dynamic programming, we shall adopt binary segmentation as our representative algorithm. When it comes to static Markov network estimation, we also have two methods —- BMN (the old method) and PL-LR, the latter of which can transform the original Markov network into the logistic regression to enhance efficiency and accuracy. To be more specific, the PL-LR method is more suitable for being inserted into the change-point detection framework compare with BMN, because PL-LR can have precise estimation within a shorter time interval, which can contribute to a more elaborate grid search for change-point detection. Therefore, we decide to choose BS-PLLR as our most representative method to be compared with the baseline TVI-FL method, which means combining the binary segmentation optimization algorithm with the PL-LR estimation method. We shall also show the BS-BMN algorithm results to illustrate that PL-LR reliably ameliorates the original BMN method of estimation for static Markov Random Field.

Following the above settings, we carry out the numerical experiment. Table 1 shows the results, in which we mainly compare the performances of three methods — BS-PLLR, BS-BMN and TVI-FL (baseline method). The graph structure follows the regular graph setting, with two variables changing which is the degree of each node d and the number of nodes p. A larger p means there is a higher-dimensional network. A larger d means there are more edges in the network, bringing a decrease of the sparsity for graph structure. For

Table 1: Results for the model with the lowest BIC, and that with the highest AUC. The average $\pm$ (std) of the metrics is reported.

Number of Nodes	**Degree**	**Method**	**BIC**		**AUC**	
			h**-score** $\downarrow$	F_1**-score** $\uparrow$	h**-score** $\downarrow$	F_1**-score** $\uparrow$
$p = 20$	$d = 2$	BS-PLLR	$\mathbf{0.005 \pm (0.002)}$	$\mathbf{0.651 \pm (0.042)}$	$\mathbf{0.004 \pm (0.002)}$	$\mathbf{0.666 \pm (0.035)}$
		BS-BMN	$0.090 \pm (0.081)$	$0.306 \pm (0.024)$	$0.067 \pm (0.041)$	$0.474 \pm (0.044)$
		TVI-FL	$0.131 \pm (0.049)$	$0.262 \pm (0.030)$	$0.190 \pm (0.017)$	$0.306 \pm (0.024)$
	$d = 3$	BS-PLLR	$\mathbf{0.003 \pm (0.003)}$	$\mathbf{0.619 \pm (0.026)}$	$\mathbf{0.004 \pm (0.003)}$	$\mathbf{0.630 \pm (0.021)}$
		BS-BMN	$0.014 \pm (0.011)$	$0.548 \pm (0.031)$	$0.064 \pm (0.055)$	$0.555 \pm (0.052)$
		TVI-FL	$0.143 \pm (0.034)$	$0.354 \pm (0.026)$	$0.184 \pm (0.031)$	$0.381 \pm (0.030)$
	$d = 4$	BS-PLLR	$\mathbf{0.003 \pm (0.002)}$	$\mathbf{0.617 \pm (0.044)}$	$\mathbf{0.003 \pm (0.002)}$	$\mathbf{0.644 \pm (0.030)}$
		BS-BMN	$\mathbf{0.003 \pm (0.002)}$	$0.599 \pm (0.024)$	$0.004 \pm (0.003)$	$0.617 \pm (0.022)$
		TVI-FL	$0.132 \pm (0.066)$	$0.371 \pm (0.019)$	$0.194 \pm (0.026)$	$0.426 \pm (0.018)$
$p = 30$	$d = 2$	BS-PLLR	$\mathbf{0.005 \pm (0.003)}$	$\mathbf{0.504 \pm (0.033)}$	$\mathbf{0.005 \pm (0.003)}$	$\mathbf{0.523 \pm (0.025)}$
		BS-BMN	$0.146 \pm (0.039)$	$0.284 \pm (0.112)$	$0.178 \pm (0.025)$	$0.319 \pm (0.068)$
		TVI-FL	$0.151 \pm (0.051)$	$0.199 \pm (0.017)$	$0.198 \pm (0.007)$	$0.215 \pm (0.017)$
	$d = 3$	BS-PLLR	$\mathbf{0.003 \pm (0.001)}$	$\mathbf{0.529 \pm (0.022)}$	$\mathbf{0.002 \pm (0.001)}$	$\mathbf{0.572 \pm (0.016)}$
		BS-BMN	$0.058 \pm (0.040)$	$0.445 \pm (0.070)$	$0.092 \pm (0.088)$	$0.470 \pm (0.068)$
		TVI-FL	$0.154 \pm (0.036)$	$0.237 \pm (0.025)$	$0.204 \pm (0.003)$	$0.269 \pm (0.017)$
	$d = 4$	BS-PLLR	$\mathbf{0.002 \pm (0.001)}$	$\mathbf{0.526 \pm (0.022)}$	$\mathbf{0.002 \pm (0.001)}$	$\mathbf{0.558 \pm (0.023)}$
		BS-BMN	$0.024 \pm (0.019)$	$0.514 \pm (0.056)$	$0.056 \pm (0.052)$	$0.540 \pm (0.045)$
		TVI-FL	$0.088 \pm (0.042)$	$0.240 \pm (0.038)$	$0.204 \pm (0.003)$	$0.313 \pm (0.014)$
$p = 40$	$d = 2$	BS-PLLR	$\mathbf{0.005 \pm (0.003)}$	$\mathbf{0.473 \pm (0.034)}$	$\mathbf{0.004 \pm (0.003)}$	$\mathbf{0.489 \pm (0.024)}$
		BS-BMN	$0.128 \pm (0.037)$	$0.257 \pm (0.084)$	$0.166 \pm (0.030)$	$0.279 \pm (0.038)$
		TVI-FL	$0.185 \pm (0.057)$	$0.130 \pm (0.023)$	$0.196 \pm (0.005)$	$0.159 \pm (0.003)$
	$d = 3$	BS-PLLR	$\mathbf{0.003 \pm (0.002)}$	$\mathbf{0.506 \pm (0.018)}$	$\mathbf{0.002 \pm (0.001)}$	$\mathbf{0.525 \pm (0.017)}$
		BS-BMN	$0.118 \pm (0.086)$	$0.278 \pm (0.080)$	$0.143 \pm (0.044)$	$0.374 \pm (0.075)$
		TVI-FL	$0.117 \pm (0.035)$	$0.202 \pm (0.006)$	$0.189 \pm (0.022)$	$0.227 \pm (0.009)$
	$d = 4$	BS-PLLR	$\mathbf{0.002 \pm (0.001)}$	$\mathbf{0.512 \pm (0.019)}$	$\mathbf{0.002 \pm (0.001)}$	$\mathbf{0.534 \pm (0.013)}$
		BS-BMN	$0.079 \pm (0.054)$	$0.456 \pm (0.052)$	$0.085 \pm (0.052)$	$0.488 \pm (0.056)$
		TVI-FL	$0.097 \pm (0.035)$	$0.214 \pm (0.023)$	$0.205 \pm (0.005)$	$0.263 \pm (0.004)$

each group, we repeat the randomized experiment ten times to get the average value and standard deviation.

Among these methods, We can see that BS-PLLR mostly dominates all the other methods, in the aspects of both h-score and F_1-score. It is worth noting that the h-score of BS-PLLR performs exceptionally well and consistently maintains an extremely precise level of change point detection however the network changes. BS-BMN also achieves good performance in the task of change point detection and graph structure estimation, and even in some cases BS-BMN has a significant level of efficacy comparable to that of BS-PLLR. However, the benchmark method TVI-FL performs mediocrely. TVI-FL's relatively high h-score suggests imprecision in change point detection, while its notably low F_1-score indicates suboptimal performance in graph structure recovery.

In addition, the results also give some insights into change point detection and graph structure recovery for networked data. When the network is higher-dimensional, F_1-score tends to be at a relatively low level, which is easily understood as higher network dimensionality implies greater difficulty for computation and estimation. When the degree of each node in the network increases, it means the sparsity of graph structure decreases, which demonstrates a larger F_1-score. This result may be because our estimation method for static network is more suitable for less sparse graphs. Another finding is that the h-score tends to decrease as d increases for BS-PLLR and BS-BMN, while TVI-FL appears seemingly impervious in the performance of change point detection. We think the reason might be that when the underlying graphs are more dense, they exhibit more pronounced time-varying signals for change point detection, which can exactly be captured by our methods instead of the benchmark TVI-FL. After vertically examining and comparing the table, we can turn to horizontal observations. It seems that the BIC tuning parameters contribute to a lower h-score, which aligns with the purpose of controlling parameters' complexity, at the cost of some loss in graph structure estimation accuracy. On the contrary, the AUC

orientation focuses on a higher F_1-score, which is in essence a more comprehensive graph structure recovery and needs to bear the rough change point detection.

We have done a quite thorough analysis on the performances from these metrics. It can be noticed that the results in Table 1 only includes the methods driven by BS (binary segmentation algorithm). In fact we also have experiments using the DP (dynamic programming algorithm), which illustrates the effects quite similar to those of BS. Thus, they are not included in the above performance table. However, although they have the analogous accuracies, their computational costs not on the same order of magnitude. We shall show the time cost for BS and DP, mainly adopting BS-PLLR and DP-PLLR owing to the remarkable performance of PLLR. We also provided the timeliness of the baseline method TVI-FL as a comparison.

Table 2: Results for the computing time of different methods. The unit for the computing time is seconds.

Number of Nodes	**Degree**	**Computing Time (s)**		
		BS-PLLR	*DP-PLLR*	*TVI-FL*
$p = 20$	$d = 2$	11.9	785.3	140.1
	$d = 3$	12.9	853.6	124.2
	$d = 4$	6.2	396.8	94.5
$p = 30$	$d = 2$	10.8	714.5	61.7
	$d = 3$	11.9	756.4	66.4
	$d = 4$	10.9	721.9	68.2
$p = 40$	$d = 2$	12.5	804.3	111.3
	$d = 3$	13.9	890.6	81.4
	$d = 4$	14.1	912.1	103.6

As shown in Table 2, we have recorded the time costs incurred during the numerical experiments. It can be observed that BS-PLLR method is the fastest among the three methods and significantly outperforms the other two in terms of time cost. DP-PLLR is

much slower than BS-PLLR, which aligns with the approximate time complexities derived based on the algorithm steps: the dynamic programming algorithm has a time complexity of $\mathcal{O}(T^2\text{Penalty}(T))$, and the binary segmentation algorithm has a time complexity of $\mathcal{O}(T\log(T)\text{Penalty}(T))$, where $\text{Penalty}(T)$ represents the time complexity of estimating the static Markov network with L_1 penalty as in Equation (2.12), involving a dataset of size T.

Furthermore, it can be observed that as the dimensionality of the model increases (i.e., when p becomes larger), the computation time also increases. Additionally, when sparsity weakens (i.e., the number of edges increases), the time required also increases. This is because, with decreasing sparsity, the graph structure has more parameters to estimate, leading to increased computational cost. When the model is simpler (smaller p), sparsity has a less pronounced effect on time cost. However, as the model becomes more complex (larger p), the differences in time cost become significantly more pronounced with decreasing sparsity.

In terms of the static Markov network structure estimation, both PLLR and BMN perform significantly better than the baseline method TVI-FL, with PLLR outperforming BMN. For change point detection, we have proposed BS and DP optimization algorithms to be combined with PLLR and BMN respectively. Both algorithms exhibit similar accuracy, but BS has a lower time cost compared to DP. Overall, considering model performance and computational cost, we conclude that BS-PLLR is the superior strategy for multiple change point detection tasks as well as the underlying graph structure recovery.

5 Application to real-world data

In this section, we aim to evaluate the practicality of our developed methods in real-world scenarios. We apply time-varying Markov random fields to model the real-world data, seeking to identify potential change points and analyze the network structure of the data

along with its corresponding real-world associations. We employ two kinds of data to model and analyze.

5.1 Voting data

Referring to (Le Bars et al., 2020), we use the voting data of the Illinois House of Representatives during the period of the 114th and 115th US Congresses[1]. The Illinois House of Representatives consists of 18 seats, each representing a district and corresponding to a US Representative from either the Democratic or Republican parties. A Representative may or may not be reelected at the end of a Congress, impacting whether they retain their seat in the new Congress. The dataset we used includes 1264 votes, each represented by a vector of size $p = 18$, with each component indicating each seats. In this vector, a dimension is 1 if the respective Representative voted Yes, and 0 if they voted No. When no information is available about a seat's vote (e.g., due to an absence), we impute the vote with the majority decision of the representative's party.

We conduct experiments on this voting dataset, modeling it as a time-varying Markov Random Fields to detect its change points. We use the BIC for hyper-parameter selection, which demonstrated the characteristic of sparse change points in numerical experiments. We choose the BS-PLLR method for our experiments, as its superior performance has been validated in numerical tests. After conducting parameter search and selection, the data exhibits a strong signal for a single change point, which occurred at the 662-th time point. Consequently, we identifies a single change point model, resulting in a two-segment Markov Random Field, specifically segments $[1, 661]$ and $[662, 1264]$.

We begin by introducing what a change point signal is and how to assess the strength of such signals. This is one of the unique advantages of the BS-PLLR method, as it can display the strength of change point signals, thereby distinguishing which change points are

[1]The data can be found at www.voteview.com.

significant with substantial structural changes and which are less clear and may fluctuate with changes in hyper-parameters.

Referring back to the binary segmentation method discussed earlier, we can see from Equations (2.18) and (2.19) that each time a potential binary split is made from a parent interval, we need to consider the sum of the losses of the two subsegments compared to the loss of the parent segment. If this difference exceeds the hyper-parameter γ, the parent interval is split; if it is less than γ, it is not. This difference can be considered the change point signal. A strong signal will drive the parent interval to split into two sub-intervals, while a weak signal will cause the parent interval to remain intact and never split. This splitting operation starts from the original interval of the whole observations and continues until no further changes occur in any interval. Therefore, the two sub-intervals resulting from the initial split of the root interval indicate the strongest change point signal. The deeper the binary tree splits, the weaker the signal, until it is weaker than the hyper-parameter γ, and the final state of the binary tree remains unchanged.

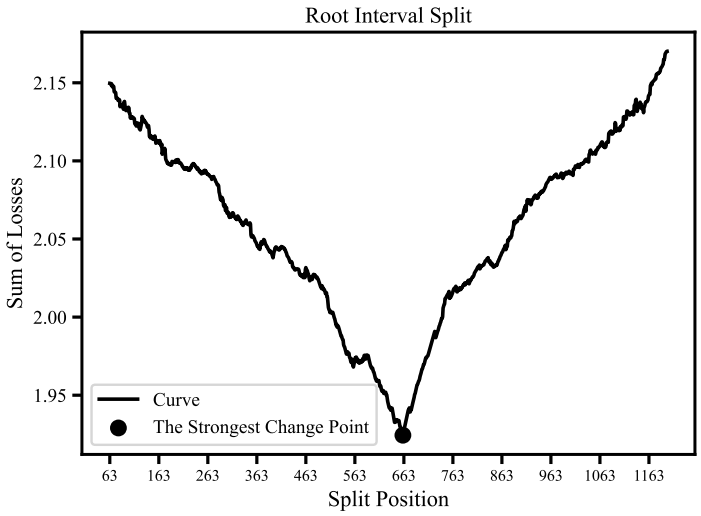

Figure 1: First split operation by the binary segmentation for the voting data

We can illustrate this splitting process of the original full interval in a diagram, as shown in Figure 1. It is the splitting result of our voting dataset. We consider the split

position as x-axis and the sum of losses as y-axis. Each pair (x, y) represents a split at point x into two segments, and y represents the sum of their loss functions (the negative log-pseudo-likelihood).. The smaller this sum, the stronger the signal for that point to become a potential change point. From Figure 1, we can see an extremely significant change point, which corresponds to the minimum of the sum of the losses for the two segments. This minimum point is precisely at 662, identifying our change point. Furthermore, no other candidate change points show any significant signals, making this the only change point after model selection by the BIC criterion. That means Figure 1 is the final result of binary splitting and change point detection. We will discuss an example involving signals for multiple change points in the following section using another kind of data. This will not be elaborated further here.

Now we know that there is only one change point in our final model, which means that there are two stages for the data. We want to know whether our detected change point can make sense. As a matter of fact, the change point occurs at the 662th time point, which coincides with the transition between the 114th Congress and the 115th Congress. At that time, the 114th Congress had just concluded and the 115th Congress was beginning. External personnel changes have reshaped the balance of interests among the various seats, leading to underlying changes in the network structure. To be more specific, we focus on the specific network structure shown in Figure 2.

There are some instructions regarding the information in the figure. There are 18 seats and we use different colors to represent different political parties. We use black edges to represent positive correlations and gray edges to represent negative correlations. Additionally, the thickness of the edges indicates the strength of the connections. We can clearly find that there are significant changes between these two stages. The 10th seat was closer to the Republican party during the first phase, but in the second phase, it developed more connections with the Democratic party. This shift is reflected not only in the number

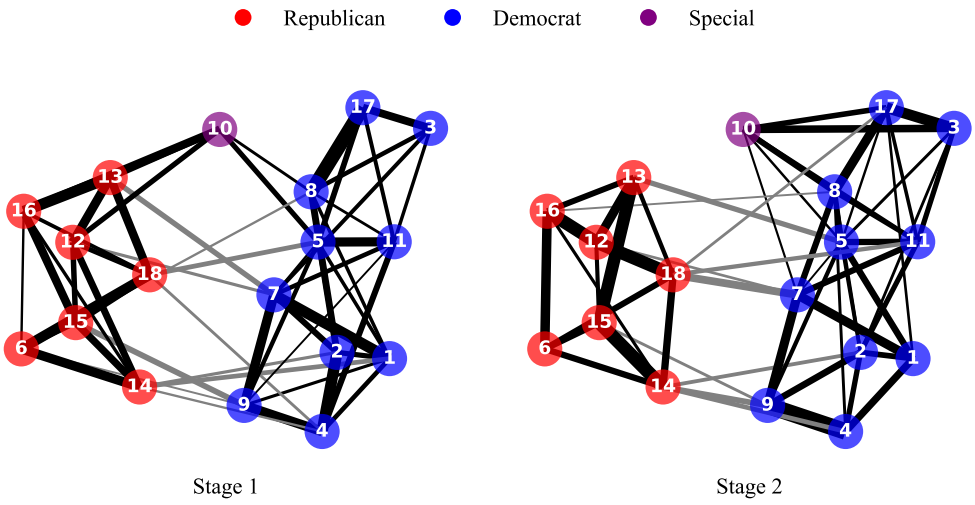

Figure 2: Network strutures for voting data

Note: The red nodes are Republicans, and the blue nodes are Democrats. The 10th node is distinctly highlighted with a different color due to its special status.

of edges in the network but also in the strength (thickness) of these edges. We can also easily observe connections between the Republican and Democratic parties, most of which are negatively correlated. Let us have a look at the 10th seat (node). In the first stage, the 10th seat seemed to play a unique role in the interactions between the Democratic and Republican parties, acting as a mediator. However in the second stage, the 10th seat was no longer swinging between the two parties and has now shifted in favor of the Democrats. In fact, during the 114th Congress, the 10th seat was held by Republican Robert Dold who was more like a collaborator, but in the 115th Congress, he was replaced by Democrat Brad Schneider.

5.2 Commodity futures data

We use the data of commodity futures prices, which is collected from the Bloomberg database[2]. It consists of 32 commodity futures in the main futures market in the United

[2]The data can be found at www.bloomberg.com/asia.

States from 1998 to 2022 on a daily frequency. We believe that these commodities may form a network, where each edge measures the price correlation, such as simultaneous increases and decreases or inverse relationships. If the price of a particular futures contract rises compared to the previous day, we assign a value of 1 to this node; if it falls, we assign a value of 0, thereby forming the time-varying Markov random fields. The specific futures name, abbreviations and sector categories are shown in the Appendix.

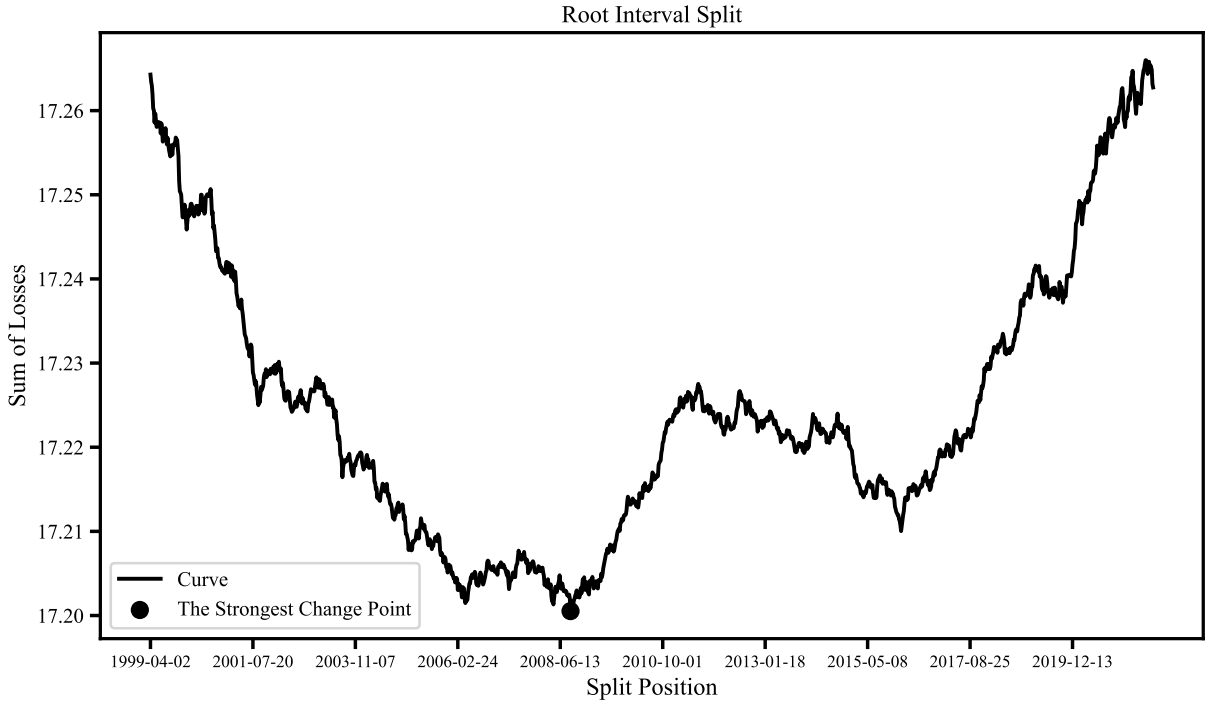

Figure 3: First split operation by the binary segmentation for the commodity futures data

Firstly we have a look at the original interval splitting result by the method BS-PLLR in Figure 3. Obviously, it shows the strong signal for potential multiple change points. This loss curve has undergone only one binary split, but it still shows several concave points, indicating the potential for further splits in subsequent binary partitioning. Another piece of information we can gather from the graph is that the lowest point represents the split point of the initial binary partition, which is also the most significant change point for the commodity futures data. As a matter of fact, the strongest change point is on September 5, 2008, which was precisely on the eve of the financial crisis.

Furthermore, we apply the complete BS-PLLR method to the futures data for detecting multiple change points. We also perform model selection using the BIC criterion to determine the optimal change point locations. The ultimate results are as follows in Figure 4. Apart from the start and end points, all the intermediate points on the axis represent the detected change points. Let us examine whether these detected change points make sense in the context of the real world. The most significant change point is on September 5, 2008, right on the eve of the official outbreak of the financial crisis. The subprime mortgage crisis was intensifying, and the real estate bubble was about to burst. Two days later, on September 7, 2008, the Federal takeover of Fannie Mae and Freddie Mac was implemented. Ten days later, Lehman Brothers declared bankruptcy, marking the official onset of the financial crisis. We shall check the remaining change points. The first change point is on September 7, 2001. During this period, the United States was experiencing an economic downturn following the burst of the dot-com bubble. This downturn was exacerbated by the 9/11 terrorist attacks, which occurred four days later, causing a sharp decline in economic activity. The next change point is August 18, 2003. The Iraq War, which began in March 2003, led to instability in the Middle East, and fluctuations in oil prices caused instability in the futures market. The change point on July 31, 2006, corresponds to the Lebanon War between Israel and Hezbollah, which lasted for over a month. The conflict led to instability in oil prices, affecting the futures market. On February 8, 2016, concerns about the slowing global economic growth, particularly the deceleration of China's economy, triggered global market volatility. The final change point in the data is April 28, 2020. It marks the beginning of the recovery from the economic collapse caused by the COVID-19 pandemic in early 2020. Just a month earlier, the U.S. stock market had experienced five circuit breakers. By the end of April, the global economy began to recover, with oil prices showing a noticeable turning point.

Based on the six change points, we divide the whole time period from 1998 to 2022 into

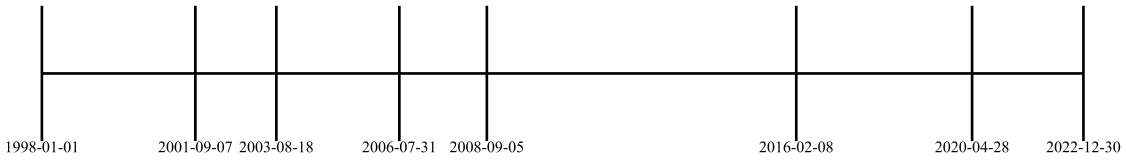

Figure 4: Multiple change points for the commodity futures data

seven stages. In each stage, the data share the same underlying graph structure. They are shown in Figure 5. We observe that the network density increased significantly after 2001 and remained elevated until 2008. This may be attributed to the ongoing geopolitical conflicts that affect futures prices, resulting in persistent volatility and sustained interdependencies between futures contracts. Following the significant turning point of the financial crisis, the network became sparse. Subsequently, as the economy slowly recovered, the network density began to rise again. However, the COVID-19 pandemic in 2020 was a great shock to the economy. The strong connection decreased and the weak connection increased. The thickness of the edges represents the strength of these connections. These futures are divided into several sector categories according to the standards shown in the Appendix. We can clearly see that connections within the same category are stronger than those between different categories. Inter-category connections tend to change over time, while most intra-category connections persist. Let us look at a few examples. The FC and LC reprensent feeder cattle and live cattle, appearing to have strong connections as the same livestock category. The same condition can be found among S, SM and BO, which represent soybean, soybean meal and soybean oil in the agriculture category. The upstream and downstream relationships within the supply chain create network connections between nodes, and these connections are persistent and exceptionally strong. However, the connection between LA(aluminium) and HG(copper) is not always significant.

We also formulate the degree for these sector categories. Refer to (Wasserman, 1994), we adopt the concept of normalized average degree centrality for groups. We firstly calculate the centrality of each node within the group and sum these values. We then normalize

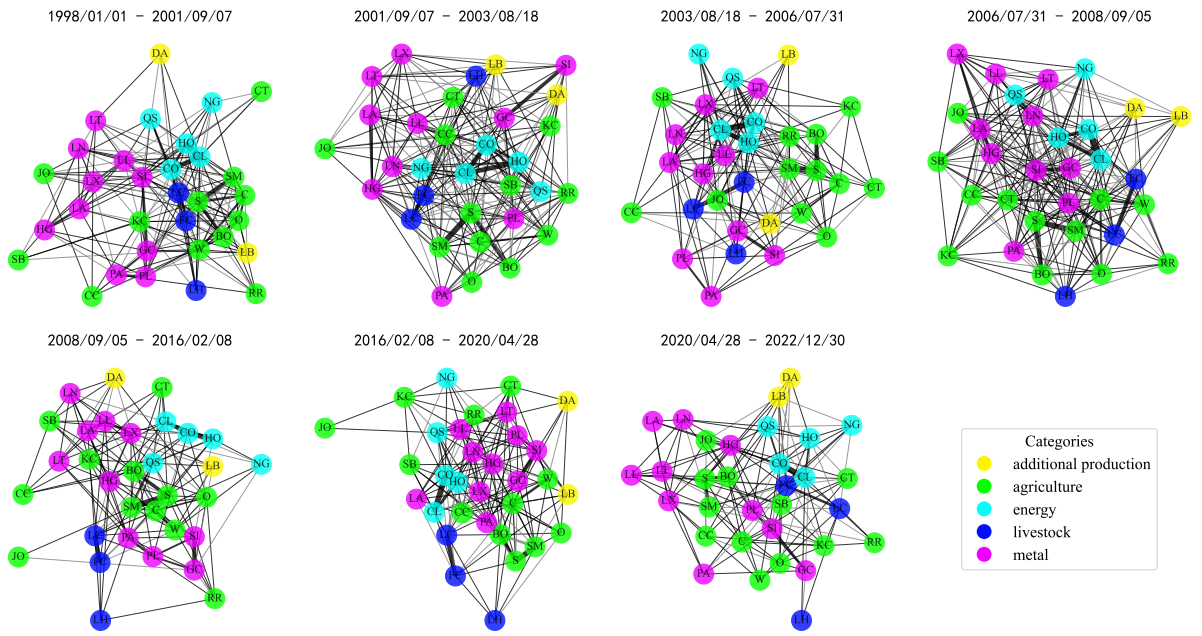

Figure 5: Network structures in different stages for futures data

this sum using the group's size and the overall scale of the network to ensure that the final computed result lies within the interval $[0, 1]$. We plot the results over the entire time period in Figure 6. We can see that over time, the centrality of the groups exhibits clear increasing or decreasing trends, and these trends are generally consistent across all groups. This variation in trends further validates our previous analysis of the network structure. Additionally, we note that the centralities of the energy and livestock groups have constantly remained at high levels, while those of the metal and agriculture groups have persistently been low. Notably, the centrality of the additional production group soared to become the highest during the period from 2020 to 2022.

6 Conclusions

The Markov Random Field is a classical probabilistic undirected graphical model commonly used to represent real-world networks. In this paper, we propose the time-varying Markov Random Fields that can model dynamic networks, detect multiple significant change points

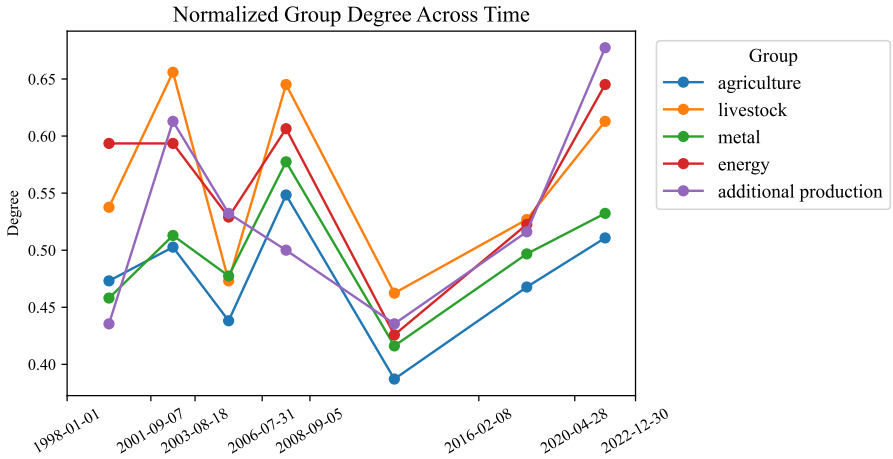

Figure 6: Group degree for futures data

within the network, and recover the underlying graph structure at each stage. We employ the pseudo-likelihood method to estimate the graph structure and introduce dynamic programming and binary segmentation algorithms to achieve optimal detection of multiple change points. Innovatively, we equivalently transform the Markov Random Field into logistic regression while preserving its temporal invariance, which significantly reduces the computational cost for estimating the graph structure and improves the accuracy of change point detection. In our numerical experiments, we compare our method with existing baseline approaches and achieve better results in both estimation accuracy and time efficiency. We apply the time-varying Markov Random Fields to model voting data and futures data, gaining insightful understanding of the evolution of internal structures within the data, thereby demonstrating the effectiveness of our proposed method.

References

Ahmed, A., and Xing, E. P. (2009), "Recovering time-varying networks of dependencies in social and biological studies," *Proc. of the National Academy of Sciences*, 106, 11878–11883.

Banerjee, O., Ghaoui, L. E., and d'Aspremont, A. (2008), "Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data," *Journal of Machine Learning Research*, 9, 485–516.

Bybee, L., and Atchadé, Y. (2018), "Change-point computation for large graphical models: a scalable algorithm for Gaussian graphical models with change-points," *Journal of Machine Learning Research*, 19, 440–477.

Fazayeli, F., and Banerjee, A. (2016), "Generalized direct change estimation in ising model structure," in *Int. Conf. on Machine Learning*, pp. 2281–2290.

Friedman, J., Hastie, T., and Tibshirani, R. (2008), "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, 9, 432–441.

Friedman, J., Hastie, T.— (2010), "Regularization paths for generalized linear models via coordinate descent," *Journal of statistical software*, 33, 1–22.

Friedrich, F., Kempe, A., Liebscher, V., and Winkler, G. (2008), "Complexity penalized M-estimation: Fast computation," *Journal of Computational and Graphical Statistics*, 17, 201–224.

Geng, S., Kuang, Z., and Page, D. (2017), "An efficient pseudo-likelihood method for sparse binary pairwise Markov network estimation," *arXiv preprint arXiv:1702.08320*.

Gibberd, A. J., and Nelson, J. D. (2017), "Regularized estimation of piecewise constant Gaussian graphical models: The group-fused graphical lasso," *Journal of Computational and Graphical Statistics*, 26, 623–634.

Gibberd, A. J., and Roy, S. (2017), "Multiple changepoint estimation in high-dimensional Gaussian graphical models," *arXiv preprint arXiv:1712.05786*.

Hallac, D., Park, Y., Boyd, S., and Leskovec, J. (2017), "Network inference via the time-varying graphical lasso," in *Proc. of the ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, ACM, pp. 205–213.

Hinton, G. E. (2002), "Training products of experts by minimizing contrastive divergence," *Neural computation*, 14, 1771–1800.

Höfling, H., and Tibshirani, R. (2009), "Estimation of sparse binary pairwise Markov networks using pseudo-likelihoods," *Journal of Machine Learning Research*, 10, 883–906.

Keshavarz, H., Michailidis, G., and Atchade, Y. (2018), "Sequential change-point detection in high-dimensional Gaussian graphical models," *arXiv preprint arXiv:1806.07870*.

Killick, R., Fearnhead, P., and Eckley, I. A. (2012), "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical Association*, 107, 1590–1598.

Kolar, M., Song, L., Ahmed, A., Xing, E. P. et al. (2010), "Estimating time-varying networks," *The Annals of Applied Statistics*, 4, 94–123.

Kolar, M., and Xing, E. P. (2012), "Estimating networks with jumps," *Electronic Journal of Statistics*, 6, 2069–2106.

Koller, D., Friedman, N., and Bach, F. (2009), *Probabilistic graphical models: principles and techniques*, MIT press.

Le Bars, B., Humbert, P., Kalogeratos, A., and Vayatis, N. (2020), "Learning the piecewise constant graph structure of a varying ising model," in *International Conference on Machine Learning*, PMLR, pp. 675–684.

Leonardi, F., and Bühlmann, P. (2016), "Computationally efficient change point detection for high-dimensional regression," *arXiv preprint arXiv:1601.03704*.

Londschien, M., Kovács, S.— (2019), "Change point detection for graphical models in presence of missing values," *arXiv preprint arXiv:1907.05409*.

Meinshausen, N., Bühlmann, P. et al. (2006), "High-dimensional graphs and variable selection with the lasso," *The Annals of Statistics*, 34, 1436–1462.

Ravikumar, P., Wainwright, M. J., Lafferty, J. D. et al. (2010), "High-dimensional Ising model selection using ℓ1-regularized logistic regression," *The Annals of Statistics*, 38, 1287–1319.

Roy, S., Atchadé, Y., and Michailidis, G. (2017), "Change point estimation in high dimensional Markov random-field models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79, 1187–1206.

Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J., and Tibshirani, R. J. (2012), "Strong rules for discarding predictors in lasso-type problems," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74, 245–266.

Truong, C., Oudre, L., and Vayatis, N. (2019), "Selective review of offline change point detection methods," *Signal Processing*, 167, 107299.

Viallon, V., Banerjee, O., Jougla, E., Rey, G., and Coste, J. (2014), "Empirical comparison study of approximate methods for structure selection in binary graphical models," *Biometrical Journal*, 56, 307–331.

Wang, B., Qi, Y. et al. (2018), "Fast and Scalable Learning of Sparse Changes in High-Dimensional Gaussian Graphical Model Structure," in *Int. Conf. on Artificial Intelligence and Statistics*, pp. 1691–1700.

Wasserman, S. (1994), "Social network analysis: Methods and applications," *The Press Syndicate of the University of Cambridge*.

Xue, L., Zou, H., Cai, T. et al. (2012), "Nonconcave penalized composite conditional likelihood estimation of sparse Ising models," *The Annals of Statistics*, 40, 1403–1429.

Yang, E., and Ravikumar, P. (2011), "On the Use of Variational Inference for Learning Discrete Graphical Model." in *ICML*, pp. 1009–1016.

Yang, J., and Peng, J. (2019), "Estimating Time-Varying Graphical Models," *Journal of Computational and Graphical Statistics*, 29, 191–202.

Yuan, M., and Lin, Y. (2007), "Model selection and estimation in the Gaussian graphical model," *Biometrika*, 94, 19–35.

Zhao, T., Liu, H., and Zhang, T. (2018), "Pathwise coordinate optimization for sparse learning: Algorithm and theory," *Annals of Statistics*, 46, 180–218.

Appendices

Table 3: Supporting information for commodity futures data

Abbreviation	**Name**	**Category**
C	Corn	agriculture
CC	Cocoa	agriculture
KC	Coffee	agriculture
CT	Cotton	agriculture
O	Oats	agriculture
JO	Orange juice	agriculture
S	Soybean	agriculture
SM	Soybean Meal	agriculture
BO	Soybean Oil	agriculture
SB	Sugar	agriculture
W	Wheat	agriculture
RR	Rough Rice	agriculture
FC	Feeder Cattle	livestock
LH	Hogs	livestock
LC	Live Cattle	livestock
LA	Aluminum	metal
HG	Copper	metal
GC	Gold	metal
LL	Lead	metal
LN	Nickel	metal
PA	Palladium	metal
PL	Platinum	metal
SI	Silver	metal
LT	Tin	metal
LX	Zinc	metal
CO	Brent Crude Oil	energy
CL	Crude Oil	energy
QS	Gas Oil	energy
NG	Natural Gas	energy
HO	Heating Oil	energy
DA	Milk	additional production
LB	Lumber	additional production