Local Low-Rank Hawkes Processes for Temporal User-Item Interactions

Jin Shang
Division of Computer Science and Engineering
Louisiana State University
Email:jshang2@lsu.edu

Mingxuan Sun
Division of Computer Science and Engineering
Louisiana State University
Email:msun@csc.lsu.edu

***Abstract*—Hawkes processes have become very popular in modeling multiple recurrent user-item interaction events that exhibit mutual-excitation properties in various domains. Generally, modeling the interaction sequence of each user-item pair as an independent Hawkes process is ineffective since the prediction accuracy of future event occurrences for users and items with few observed interactions is low. On the other hand, multivariate Hawkes processes (MHPs) can be used to handle multi-dimensional random processes where different dimensions are correlated with each other. However, an MHP either fails to describe the correct mutual influence between dimensions or become computational inhibitive in most real-world events involving a large collection of users and items. To tackle this challenge, we propose local low-rank Hawkes processes to model large-scale user-item interactions, which efficiently captures the correlations of Hawkes processes in different dimensions. In addition, we design an efficient convex optimization algorithm to estimate model parameters and present a parallel algorithm to further increase the computation efficiency. Extensive experiments on real-world datasets demonstrate the performance improvements of our model in comparison with the state of the art.**

***Index Terms*—Hawkes Process; Kernel Smoothing; Sequential Data**

I. Introduction

Hawkes processes have become very popular in modeling recurrent user-item interaction events that exhibit mutual-excitation properties in various domains [1], [2]. For example, Hawkes processes can be used to model user behaviors in online services, where the interaction of a user with an item such as visiting a website or watching a movie may trigger future interactions with other correlated items. Recent approaches [3], [4] treat the event occurrences of each user-item pair as a point process and predict the next occurrence of user-item interaction based on previous interactions. Accurate modeling user-item interactions may have significant economic impact on online platforms such as revenue boost due to targeted advertising.

Formally, the Hawkes process for modeling an interaction sequence of a single user-item pair $(u,\ i)$ can be characterized by parameters such as a base intensity and a self-exciting coefficient that captures the influence of each previous event. Intuitively, m-by-n Hawkes processes can be used to model interaction sequences for m users and n items, where the base intensities and the self-exciting coefficients are represented as m-by-n matrices, respectively. Since users and items can usually be grouped into a limited number of clusters, we can assume that each parameter matrix has a low-rank structure. However, the prediction accuracy of future event occurrences for users and items with few observed interactions is low since the point processes are independent of each other [3]. In fact, only a few recurrent events such as purchases are observed for a majority of pairs of users and items in many large-scale real-world scenarios.

One way to alleviate the cold-start issue is to incorporate auxiliary features such as user demographics and item content features. For example, a coevolutionary model [5] takes advantage of auxiliary features such as item genres and incorporates the former events of all user-item pairs with different weights. The time prediction performance has been improved since more data are used to fit the model parameters of each user-item pair, but the item prediction performance has decreased due to the combination of the events from all user-item pairs.

On the other hand, a multivariate Hawkes process (MHP) [6], [7] can be used to handle a multi-dimensional (e.g., $N = m \times n$) random process where different dimensions are correlated with each other. Specifically, the conditional intensity for the i-th dimension is characterized by the base intensity and the linear combination of the influences of events occurred in every other dimension on the i-th dimension. Extensive research [8]–[11] has focused on estimating the $N \times N$ excitation matrix of a multivariate process for various inference tasks. However, an MHP either fails to describe the correct mutual influence between dimensions or becomes computational expensive in most real-world applications involving a large collection of event sequences [10], [12], [13].

In this paper, we propose local low-rank Hawkes processes to model large-scale user-item interactions, which efficiently captures the correlations of Hawkes processes in different dimensions. Specifically, a Hawkes process is used to model the interaction sequence of each user-item pair. The parameter matrix for all processes, such as the base intensity matrix and self-exciting coefficient matrix, is assumed to behave as a low-rank matrix in the neighborhoods of certain user-item combinations. Each parameter matrix is expressed as a smoothed aggregation of several low-rank matrices, which approximates the parameters in a local neighborhood. We adopt non-parametric kernel smoothing to aggregate several local models into a unified model approximation. Based on the local

low-rank approximation, the Hawkes processes for all user-item pairs are correlated due to the similarities between local mappings of the parameter matrices. In addition, we design an efficient convex optimization algorithm to estimate model parameters and present a parallel algorithm to further increase the computation efficiency. Extensive experiments on real-world datasets demonstrate the performance improvements of our model in comparison with the state of the art.

II. Related Work

Local low-rank matrix completion with kernel smoothing [14] has been applied to matrix factorization, in which the observed ratings are formulated as a matrix and simulated with several local mappings. Each local mapping is assumed to be low-rank and the missing ratings are reconstructed with a non-parametric regression of those mappings. Previous work [14] mainly focuses on two-dimensional matrix factorization without the temporal dimension. Our work is modeling a sequence of events and the objective function is completely different from the mean squared loss of ratings in matrix completion tasks. In addition, we use the trace norm to enforce the low-rank assumption and the previous work [14] explicitly decomposes a matrix to the product of two low-rank matrices. Variations of matrix completion methods [15]–[17] are widely applied in recommender systems.

Hawkes processes [6] can be used in a variety of applications such as inferring granger causality [11], modeling patient records in smart health [18], and predicting online social activities [1]. For example, a multi-dimensional Hawkes process has been proposed by Zhou et al. [1] to learn the social event diffusion in sparse low-rank networks. A multivariate Hawkes process has further been proposed by Farajtabar et al. [2] to capture both endogenous and exogenous event intensities in social network events. Limitations of the multivariate Hawkes process such as the computational inefficiency for modeling real-world events have been studied in [10], [12], [13].

For modeling large-scale user-item interactions, previous work [3] simulates the temporal events of (u, i) pair as a one-dimensional Hawkes process and assumes that all user-item pairs are independent. The intensity of each Hawkes process is estimated based solely on the individual pair's observed sequence. The performance of the method degrades when there are no sufficient observed events for an individual (u, i) pair. We consider the approach a *point to point* intensity estimation.

A coevolutionary latent feature process has been proposed in [5], which constructs interdependent Hawkes processes by integrating additional features such as item features, user features, and interaction features between users and items. The intensity of the events of each user-item pair is estimated by aggregating the influences of all previous events of other user-item pairs weighted by user and item similarities. Hence, the time prediction accuracy improves since a large number of events are used to simulate only one pair's intensity and a huge amount of auxiliary feature information is incorporated. However, the item rank prediction becomes worse because the individual preferences are influenced by the general preference. We consider the approach a *matrix to point* intensity estimation.

Our model is different from others in that we assume a local low-rank structure in user-item dimension, which models the intensity for one (u, i) pair as the aggregation of several neighbors. The smoothing kernels are used to evaluate the similarities between these neighbors and the target (u, i) pair. The final estimation of the target (u, i) pair's intensity integrates the influences of neighbors with different weights, which we consider a *neighbors to point* intensity estimation.

III. Model

In this section, we introduce the local low-rank Hawkes process.

A. Background on Hawkes Process

A temporal point process is a random process [19], [20] and the realization of the process consists of a list of discrete temporal events $\mathcal{T} = \{t_i\}_{i=1}^n$. It is basically a counting process that counts the cumulative number of events $\{N(t), t \geq 0\}$ occurring right before time t. A counting process is also a submartingale, *i.e.*, $\mathbb{E}[N(t)|\mathcal{T}_{t'}] \geq N(t')$ for all $t > t'$, where $\mathcal{T}_{t'} = \{t_i | t_i < t'\}_{i=1}^n$ denotes the history up to but not including time t'. A temporal point process can be characterized by the conditional intensity function $\lambda(t)$, which models the occurrence of the next event given all the previous events.

The functional form of the intensity function characterizes the temporal point process. For example, the intensity of a homogeneous Poisson process is constant over time, *i.e.*, $\lambda(t) = \eta \geq 0$. Alternatively, the Hawkes process, a conditional Poisson process, is particularly useful for modeling the mutual excitation between events. For example, the intensity can be defined as:

$$\lambda(t) = \eta + \alpha \sum_{t_i \in \mathcal{T}_t} \kappa_\sigma(t - t_i), \tag{1}$$

where $\kappa_\sigma(t) := \exp(-t/\sigma)$ is an exponential kernel function capturing temporal dependencies, $\eta \geq 0$ is a base intensity capturing the long-term incentive to generate events, and $\alpha \geq 0$ is the coefficient that magnifies the influence of each previous event.

Given a collection of events between m users and n items, the occurrences of user u's interaction events with item i can be modeled as a self-exciting Hawkes process [6], *i.e.*:

$$\lambda_{(u,i)}(t) = \boldsymbol{H}_{u,i} + \boldsymbol{A}_{u,i} \sum_{t_j^{u,i} \in \mathcal{T}_t^{u,i}} \kappa_\sigma(t - t_j^{u,i}), \tag{2}$$

where $\boldsymbol{H}$ denotes an $m \times n$ matrix with the (u, i)-th entry equal to the non-negative base intensity for user-item pair (u, i), and $\boldsymbol{A}$ denotes an $m \times n$ matrix with the (u, i)-th entry equal to the self-exciting coefficient for user-item pair (u, i). The sequence $\mathcal{T}_t^{u,i} = \{t_j^{u,i} | t_j^{u,i} < t\}_{j=1}^n$ denotes the set of historic events induced between user u and item i up to but not including time t. In traditional approaches [3], [4], the two parameter matrices $\boldsymbol{H}$ and $\boldsymbol{A}$ are assumed to have low-rank structures.

A univariate Hawkes process can be extended to a multivariate Hawkes process [6], [7] to handle a multi-dimensional (e.g., $m \times n$) random process where different dimensions are correlated with each other. However, in most real-world events involving large dimensions m and n, the parameter estimation of an MHP becomes inefficient [10], [12], [13].

B. Local Low-Rank Hawkes Process

Assuming that the mapping from user-item pairs to parameters is slowly varying, the parameter matrices $\boldsymbol{H}$ and $\boldsymbol{A}$ for all user-item pairs $s = (u, i) \in [m] \times [n]$ can be characterized by a smoothed combination of multiple low-rank matrices in a way similar to [14]. Specifically, we assume that there exists a metric over the user-item space $[m]\times[n]$. The distance between pair $s_1 = (a_1, b_1)$ and pair $s_2 = (a_2, b_2)$ is denoted by $d(s_1, s_2) = d((a_1, b_1), (a_2, b_2))$, which reflects the similarity between rows a_1 and a_2 and columns b_1 and b_2. We assume that there is a set of $q < m \cdot n$ anchor user-item pairs and each of them is associated with a base intensity matrix $\boldsymbol{H}^{s_\tau}$ and a self-exciting coefficient matrix $\boldsymbol{A}^{s_\tau}$, $\tau = 1, 2, ..., q$. If $d(s_1, s_2)$ is small, $\boldsymbol{H}^{s_1}$ and $\boldsymbol{A}^{s_1}$ are similar to $\boldsymbol{H}^{s_2}$ and $\boldsymbol{A}^{s_2}$, respectively, by their spatial proximity in the embedding $\mathbb{R}^{m\times n}$. Typically, for an anchor pair $s_\tau = (a_\tau, b_\tau) \in [m]\times[n]$, the neighborhood $\{s' : d(s_\tau, s') < h\}$ in the original matrices $\boldsymbol{H}$ and $\boldsymbol{A}$ can be approximated by the corresponding entries of matrices $\boldsymbol{H}^{s_\tau}$ and $\boldsymbol{A}^{s_\tau}$.

Furthermore, we recover the mapping parameter matrices $\boldsymbol{H}$ and $\boldsymbol{A}$ from aggregating a set of $q < m \cdot n$ matrices without imposing a specific function form. Following common non-parametric approaches, we define a smoothing kernel $K_h(s_1, s_2)$, $s_1, s_2 \in [m] \times [n]$ for user-item pairs, which is a non-negative symmetric unimodal function parameterized by a bandwidth parameter $h > 0$. There are many popular choices of smoothing kernels, such as the Gaussian Kernel, Logistic Kernel, Sigmoid Kernel, and Silverman Kernel, defined as follows, respectively:

$$K_h(s_1, s_2) \propto \exp(-\frac{1}{2}h^{-2}d(s_1, s_2)^2), \tag{3}$$

$$K_h(s_1, s_2) \propto \frac{1}{\exp(d(s_1, s_2)/h) + 2 + \exp(-d(s_1, s_2)/h)}, \tag{4}$$

$$K_h(s_1, s_2) \propto \frac{1}{\exp(d(s_1, s_2)/h) + \exp(-d(s_1, s_2)/h)}, \tag{5}$$

$$K_h(s_1, s_2) \propto \exp(-\frac{|d(s_1, s_2)/h|}{\sqrt{2}}) \cdot \sin(\frac{|d(s_1, s_2)/h|}{\sqrt{2}} + \frac{\pi}{4}). \tag{6}$$

We adopt a type of locally constant kernel regression [21] to aggregate multiple local matrices. For simplicity, we use the same smoothing kernel and the same bandwidth for base intensity η and coefficient α. That is, for each user-item pair $s = (u, i)$, the occurrences of user u's interactions with item i are modeled as a local low-rank Hawkes process with the following intensity:

$$\lambda_s(t) = \sum_{\tau=1}^{q} \frac{K_h(s_\tau, s)}{\sum_{k=1}^{q} K_h(s_k, s)} [\boldsymbol{H}_s^{s_\tau} + \boldsymbol{A}_s^{s_\tau} \sum_{t_j^s \in \mathcal{T}_t^s} \kappa_\sigma(t - t_j^s)], \tag{7}$$

where $\boldsymbol{H}_s^{s_\tau}$ and $\boldsymbol{A}_s^{s_\tau}$ are the s-th entry of the τ-th base intensity matrix $\boldsymbol{H}^{s_\tau}$ and self-exciting matrix $\boldsymbol{A}^{s_\tau}$, $\tau = 1, 2, ..., q$, respectively. Note that we have matrix index $s = (u, i)$. Since the users and the items in each matrix can be grouped into a limited number of sets with similar types, we assume that $\boldsymbol{H}^{s_\tau}$ and $\boldsymbol{A}^{s_\tau}$ have low-rank structures. This means that the nuclear norms of the parameter matrices, $\|\boldsymbol{H}^{s_\tau}\|_*$ and $\|\boldsymbol{A}^{s_\tau}\|_*$, are small. Therefore, the mapping parameter matrices $\boldsymbol{H}$ and $\boldsymbol{A}$ in eq. (2) have local low-rank structures, and the local low-rank Hawkes process in eq. (7) is actually based on the weighted summation of q low-rank Hawkes processes in eq. (2). Specifically, our local low-rank Hawkes model is equivalent to the low-rank Hawkes model [3] when the number of anchor points is equal to one, i.e., $q = 1$.

To simplify the notation, we denote by $K_h^{(a,b)}$ the matrix whose (i, j)-entry is $K_h((a, b), (i, j))$. Given a series of anchor points $s_\tau \in 1, ..., q$, let $\sum_{k=1}^{q} K_h(s_k, s) = C_s$, the denominator of which is the summation of the kernel weights and actually a constant for each (u, i) pair. We further create three block matrices $\boldsymbol{H}'$, $\boldsymbol{K}'$, and $\boldsymbol{A}' \in \mathbb{R}^{m\times(q*n)}$ by concatenating a set of matrices $\boldsymbol{H}^{s_\tau}$, $K_h^{s_\tau}$, and $\boldsymbol{A}^{s_\tau}$ as follows:

$$\begin{aligned} \boldsymbol{H}' &= [\boldsymbol{H}^{s_1}, ..., \boldsymbol{H}^{s_q}], \\ \boldsymbol{A}' &= [\boldsymbol{A}^{s_1}, ..., \boldsymbol{A}^{s_q}], \\ \boldsymbol{K}' &= [K_h^{s_1}, ..., K_h^{s_q}]. \end{aligned} \tag{8}$$

Let $\boldsymbol{M}\{u, i\}$ be a vector extracted from a matrix $\boldsymbol{M}$ for each (u, i) pair, i.e., $[\boldsymbol{M}^{s_1}(u, i), ..., \boldsymbol{M}^{s_q}(u, i)]$, where $\boldsymbol{M}$ can be any of the three matrices $\boldsymbol{H}'$, $\boldsymbol{A}'$, and $\boldsymbol{K}'$.

C. Objective Function

Based on the survival analysis theory [20], the likelihood of observing a sequence of events $\mathcal{T} = \{t_i\}_{i=1}^n$ is $\prod_{t_i \in \mathcal{T}} \lambda(t_i) \cdot \exp(-\int_0^T \lambda(\tau)d(\tau))$, where T is the total observation time. Specifically, let $\mathcal{T}^{u,i}$ be the set of interaction events between entities u and i. The log-likelihood of observing each sequence $\mathcal{T}^{u,i}$ is:

$$\mathcal{L}(\mathcal{T}^{u,i} \mid \boldsymbol{X}) = \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \log(\boldsymbol{X}_{u,i}^\top \Phi_j^{u,i}) - \boldsymbol{X}_{u,i}^\top \Psi^{u,i}, \tag{9}$$

where:

$$\begin{aligned}\boldsymbol{X}_{u,i} =&(\boldsymbol{H}'\{u,i\}, \boldsymbol{A}'\{u,i\})^\top,\\ \Phi_j^{u,i} =&C_{u,i}^{-1}(\boldsymbol{K}'\{u,i\}\cdot 1,\\ &\boldsymbol{K}'\{u,i\}\cdot \sum_{t_k^{u,i}<t_j^{u,i}} \kappa_\sigma(t_j^{u,i}-t_k^{u,i}))^\top,\\ \Psi^{u,i} =&C_{u,i}^{-1}(\boldsymbol{K}'\{u,i\}\cdot T,\\ &\boldsymbol{K}'\{u,i\}\cdot \sum_{t_j^{u,i}\in\mathcal{T}^{u,i}} \int_{t_j^{u,i}}^{T} \kappa_\sigma(t-t_j^{u,i})dt)^\top. \end{aligned} \tag{10}$$

As a result, the log-likelihood of observing all user-item interaction sequences $\mathcal{O} = \{\mathcal{T}^{u,i}\}_{u,i}$ is a summation of terms by $\mathcal{L}(\mathcal{O}) = \sum_{\mathcal{T}^{u,i}\in\mathcal{O}} \mathcal{L}(\mathcal{T}^{u,i})$. We can obtain the model parameters $\boldsymbol{X}$ by minimizing the following objective function:

$$\begin{aligned} OPT = \min_{\boldsymbol{X}} -\frac{1}{|\mathcal{O}|}\sum_{\mathcal{T}^{u,i}\in\mathcal{O}} \mathcal{L}(\mathcal{T}^{u,i} \mid \boldsymbol{X}) + h(\boldsymbol{X})\\ s.t.\ \boldsymbol{X}\geq \boldsymbol{0}, \end{aligned} \tag{11}$$

where $h(\boldsymbol{X}) = \lambda\|\boldsymbol{H}'\|_* + \beta\|\boldsymbol{A}'\|_*$, $\boldsymbol{X} = [\boldsymbol{H}'; \boldsymbol{A}']$, and λ and β control the trade-off between the constrains. The nuclear norm $\|\cdot\|_*$ is a summation of all singular values and it can be used as a convex surrogate for the matrix rank [22]. Thus, minimizing $\|\boldsymbol{H}'\|_*$ and $\|\boldsymbol{A}'\|_*$ ensures each $\boldsymbol{H}^{s_\tau}$ and $\boldsymbol{A}^{s_\tau}$ to be low-rank. After obtaining $\boldsymbol{X}$, we can use eq. (7) to compute the intensity.

IV. Parameter Estimation

We propose to learn the parameters using an efficient framework. We first introduce the kernel function calculation. To learn the parameters that optimize the objective in eq. (11), we then introduce the latest Primal Averaging Conditional Gradient (PA-CndG) algorithm [23] based on the Proximal Gradient (PG) method [24], [25].

A. Kernel Calculation and Anchor Point Selection

A general kernel function is denote by $K_h(s_1, s_2)$, where $s_1, s_2 \in [m]\times[n]$. Similar to [14], we assume a product form $K_h((a_1,b_1),(a_2,b_2)) = K_{h_1}(a_1,a_2)\cdot K'_{h_2}(b_1,b_2)$, where those two kernels are on the spaces $[m]$ and $[n]$, respectively. As we have the log function in eq. (9), we use the Gaussian kernel in eq. (3) for both K and K', as it does not give zero values. Then the kernel function matrix for one anchor point can be expressed as $K_h^{s_\tau} = K_{h_1}^{a_\tau}\cdot K_{h_2}^{b_\tau} \in [m]\times[n]$, where $\tau = 1, ..., q$. The distance d in eq. (3) can be defined using affiliated information describing the similarities between the rows (users) or the columns (items). For example, using DNN models we can extract features from reviews for items [26]. If no such information is available, we can compute d based on the partially observed user-item interaction matrix. Specifically, we may convert the user-item interactions into a matrix X, where each entry indicates the frequency of item consumptions of a user during a time period. We can then compute the distances between row vectors (for users) and between column vectors (for items) using standard distance measures such as cosine similarity. When the matrix X is very sparse, we follow conventions as reported in [14] to factorize the matrix using standard incomplete SVD [27] and then compute the cosine distances between the rows and columns of factor matrices.

For choosing anchor points $s_1, ..., s_q$, we sample them uniformly from the observed (u, i) entries. It is worth mentioning that the anchor points can be selected by other strategies such as pre-cluster processing, that is clustering the (u, i) pairs into q clusters and then selecting one anchor point from each cluster. There are several clustering methods such as K-means and spectral clustering. For K-means, the input features for each (u, i) pair could be the concatenated user and item latent features obtained by SVD. For spectral clustering, the input similarity matrix is the one calculated by smoothing kernels. In our experiments, we found no significant difference between those methods empirically. As clustering is computationally expensive, we randomly select anchor points from the observed matrix in our algorithm.

B. Approximate Function and Gradient Update

Directly solving the objective in eq. (11) is difficult because the nonnegative constraints are coupled together with the non-smooth nuclear norm. To tackle the difficulties, we approximate eq. (11) by adopting a penalty method [3], [5]. Given $\rho > 0$, we introduce an auxiliary variable $\boldsymbol{Z} = [\boldsymbol{Z}_1; \boldsymbol{Z}_2]$ with the squared Frobenius norm, which leads to the new formulation in eq. (12):

$$\begin{aligned}\widehat{OPT} = \min_{\boldsymbol{X},\boldsymbol{Z}} -\frac{1}{|\mathcal{O}|}\sum_{\mathcal{T}^{u,i}\in\mathcal{O}} \mathcal{L}(\mathcal{T}^{u,i}\,|\,\boldsymbol{X}) + h(\boldsymbol{Z}) + g(\boldsymbol{X},\boldsymbol{Z})\\ s.t.\ \boldsymbol{X}\geq\boldsymbol{0}, \end{aligned} \tag{12}$$

where $g(\boldsymbol{X},\boldsymbol{Z}) = \rho\|\boldsymbol{H}' - \boldsymbol{Z}_1\|_F^2 + \rho\|\boldsymbol{A}' - \boldsymbol{Z}_2\|_F^2$. In this formulation of eq. (12), the nuclear norm regularization terms and the non-negativity constraints are handled separately. The approximate objective can always be the upper bound of the real objective given the bounded ρ [3]. For notational simplicity, we set:

$$f(\boldsymbol{X},\boldsymbol{Z}) = -\frac{1}{|\mathcal{O}|}\sum_{\mathcal{T}^{u,i}\in\mathcal{O}} \mathcal{L}(\mathcal{T}^{u,i}\mid\boldsymbol{X}) + g(\boldsymbol{X},\boldsymbol{Z}), \tag{13}$$

and the objective function becomes:

$$\begin{aligned}\widehat{OPT} = F(\boldsymbol{X},\boldsymbol{Z}) = f(\boldsymbol{X},\boldsymbol{Z}) + h(\boldsymbol{Z})\\ s.t.\ \boldsymbol{X}\geq\boldsymbol{0}, \end{aligned} \tag{14}$$

Note that $f(\cdot)$ is convex and Lipschitz continuous gradient (L-smooth), and $h(\cdot)$ is convex.

As shown in algorithm 1, we apply gradient update for model parameters X and Z in each iteration and keep three interdependent sequences U^k, X^k, and Y^k based on the schema in [25]. Specifically, we directly compute the proximal operator for $\boldsymbol{X}$ with the constraint in algorithm 1 as:

$$\begin{aligned}\boldsymbol{U}_1^k =& \arg\min_{\boldsymbol{U}_1^k\geq 0}\{\frac{1}{2\xi_k}\|\boldsymbol{U}_1^k - (\boldsymbol{Y}_1^{k-1} - \xi_k\nabla_1 f(\boldsymbol{Y}_1^{k-1},\boldsymbol{Y}_2^{k-1}))\|^2\}\\ =& (\boldsymbol{Y}_1^{k-1} - \xi_k\nabla_1(f(\boldsymbol{Y}_1^{k-1},\boldsymbol{Y}_2^{k-1})))_+. \end{aligned} \tag{15}$$

Note that $h(\boldsymbol{Z})$ only has variable $\boldsymbol{Z}$, so $h(\cdot) = 0$, which means that it is just normal Projected Gradient Descent (PGD).

Besides, $(\cdot)_+$ in Algorithm 1 sets the negative coordinates to zero.

For $\boldsymbol{Z}$, we do not directly calculate using eq. (15). Instead, we use a local linear expansion to approximate it, which is known as conditional gradient. Specifically, it differs from traditional conditional gradient method in the way that the search direction $\nabla_2 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1})$ is defined. It can be viewed as a variant of Nesterov's method [25] and is obtained by replacing the prox-mapping with a simpler linear expansion:

$$\boldsymbol{U}_2^k = arg\min_{\boldsymbol{Z}}\{\langle \nabla_2 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}), \boldsymbol{Z}\rangle + h(\boldsymbol{Z}). \tag{16}$$

Specifically, this part can be solved by first calculating the top singular vector pairs of $-\nabla_2 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1})$ and then using a line search to produce a scaling factor [3], [28].

Algorithm 1: Local Low-Rank Hawkes

Input: All the training events $\mathcal{O} = \{\mathcal{T}^{u,i}\}_{u,i}$; learning rate ξ_k; parameters ρ, λ, β; number of anchor points q; kernel function $K(\cdot)$ of widths h_1, h_2; step size $\gamma_k \in [0, 1]$;

Output: $\boldsymbol{X} = [\boldsymbol{H}'; \boldsymbol{A}']$, which is the block matrix

for $\tau = 1 \to q$ **do**
- $(a_\tau, b_\tau) :=$ a random selected (u, i) pair;
- **for** $i = 1 \to m$ **do**
  - $K_{h_1}^{a_\tau}(i) := \exp(-\frac{1}{2}h^{-2}d(a_\tau, i)^2)$;
- **end**
- **for** $j = 1 \to n$ **do**
  - $K_{h_2}^{b_\tau}(i) := \exp(-\frac{1}{2}h^{-2}d(b_\tau, j)^2)$;
- **end**

end

Choose to initialize $\boldsymbol{U}_1^0$;

Set $\boldsymbol{X}^0 = \boldsymbol{Z}^0 = \boldsymbol{U}_1^0 = \boldsymbol{U}_2^0$;

for *k ← 1 to MaxIter* **do**
- Set $\boldsymbol{Y}_1^{k-1} = (1-\gamma_k)\boldsymbol{X}^{k-1} + \gamma_k \boldsymbol{U}_1^{k-1}$;
- Set $\boldsymbol{Y}_2^{k-1} = (1-\gamma_k)\boldsymbol{Z}^{k-1} + \gamma_k \boldsymbol{U}_2^{k-1}$;
- Compute the proximal operator for $\boldsymbol{X}$:
- $\boldsymbol{U}_1^k = (\boldsymbol{Y}_1^{k-1} - \xi_k \nabla_1 (f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1})))_+$;
- Use a local linear expansion of f for $\boldsymbol{Z}$:
- $\boldsymbol{U}_2^k = arg\min_{\boldsymbol{Z}}\{\langle \nabla_2 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}), \boldsymbol{Z}\rangle + h(\boldsymbol{Z})\}$;
- Set $\boldsymbol{X}^k = (1-\gamma_k)\boldsymbol{X}^{k-1} + \gamma_k \boldsymbol{U}_1^k$;
- Set $\boldsymbol{Z}^k = (1-\gamma_k)\boldsymbol{Z}^{k-1} + \gamma_k \boldsymbol{U}_2^k$;

end

C. Convergence Analysis

For PGD method, the algorithm achieves the well-known optimal rate $O(1/k)$, i.e., a rate of $O(1/\epsilon)$ given learning rate $\xi_k \leq 1/L$, and for PA-CndG method, it also reaches $O(1/k)$ given the step size policy (1): $\gamma_k = \frac{2}{k+1}$ or (2): $\gamma_k = \arg\min_{\gamma\in[0,1]} f((1-\gamma)\boldsymbol{X}^{k-1} + \gamma \boldsymbol{U}_1^k, (1-\gamma)\boldsymbol{Z}^{k-1} + \gamma \boldsymbol{U}_2^k)$ [23]. Generally, the algorithm should still reach the optimal rate $O(1/k)$ by properly choosing the step size parameter and the learning rate. We have the convergence results for algorithm 1 in theorem 1, followed by the proof.

Theorem 1. *Let $\{\boldsymbol{Z}^k\}$, $\{\boldsymbol{X}^k\}$, $\{\boldsymbol{U}_1^k\}$, and $\{\boldsymbol{U}_2^k\}$ be the sequences generated by algorithm 1 with step size $\gamma_k = \frac{2}{k+1}$ and learning rate $\xi_k \leq 1/L$. Then we have:*

$$F(\boldsymbol{X}^k, \boldsymbol{Z}^k) - F^* \leq \frac{5LD_{max}^2}{k+1}, \tag{17}$$

where L is the Lipschitz constant of $\nabla f(x, z)$.

Proof. Define:

$$\begin{aligned} l_f(x, z; y_1, y_2) = f(x, z) &+ \langle \nabla_1 f(x, z), y_1 - x\rangle \\ &+ \langle \nabla_2 f(x, z), y_2 - z\rangle. \end{aligned} \tag{18}$$

For $\boldsymbol{X}, \boldsymbol{Z} \in \Omega$, f is Lipschitz continuous gradient and:

$$f(y_1, y_2) \leq l_f(x, z; y_1, y_2) + \frac{L}{2}\|y_1 - x\|^2 + \frac{L}{2}\|y_2 - z\|^2. \tag{19}$$

First note that:

$$\begin{aligned} \boldsymbol{X}^k - \boldsymbol{Y}_1^{k-1} &= \gamma_k(\boldsymbol{U}_1^k - \boldsymbol{U}_1^{k-1}) \\ \boldsymbol{Z}^k - \boldsymbol{Y}_2^{k-1} &= \gamma_k(\boldsymbol{U}_2^k - \boldsymbol{U}_2^{k-1}). \end{aligned} \tag{20}$$

Hence, using the definitions of $\boldsymbol{X}^k$ and $\boldsymbol{Z}^k$ in algorithm 1, we have:

$$\begin{aligned} f(\boldsymbol{X}^k, \boldsymbol{Z}^k) &\leq l_f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}; \boldsymbol{X}^k, \boldsymbol{Z}^k) \\ &\quad + \frac{L}{2}\|\boldsymbol{X}^k - \boldsymbol{Y}_1^{k-1}\|^2 + \frac{L}{2}\|\boldsymbol{Z}^k - \boldsymbol{Y}_2^{k-1}\|^2 \\ &= (1-\gamma_k) l_f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}; \boldsymbol{X}^{k-1}, \boldsymbol{Z}^{k-1}) \\ &\quad + \gamma_k l_f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}; \boldsymbol{U}_1^k, \boldsymbol{U}_2^k) \\ &\quad + \frac{L}{2}\gamma_k^2\|\boldsymbol{U}_1^k - \boldsymbol{U}_1^{k-1}\|^2 + \frac{L}{2}\gamma_k^2\|\boldsymbol{U}_2^k - \boldsymbol{U}_2^{k-1}\|^2. \end{aligned} \tag{21}$$

For simplicity, define the Bregman divergence $D(x, x') = \|x - x'\|^2$. From eq. (15), we know it is actually PGD method with $f(\cdot)$ as Lipschitz continuous gradient and constrained to convex set Ω. Based on the definition of the convex hull and the property of PGD, we have the following property:

$$\begin{aligned} \langle \boldsymbol{U}_1 - \boldsymbol{Y}_1^k, (\boldsymbol{Y}_1^{k-1} - \xi_k \nabla_1 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1})) - \boldsymbol{Y}_1^k\rangle \leq 0, \\ \forall\ \boldsymbol{U}_1 \in \Omega. \end{aligned} \tag{22}$$

Using eq. (22) and the definition of $\boldsymbol{U}_2^k$ in algorithm 1, we have:

$$\begin{aligned} &\langle \nabla_1 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}), \boldsymbol{Y}_1^k - \boldsymbol{U}_1^*\rangle \leq \\ &-\frac{1}{2\xi_k}\boldsymbol{D}(\boldsymbol{Y}_1^k, \boldsymbol{Y}_1^{k-1}) + \frac{1}{2\xi_k}[\boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^{k-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*, \boldsymbol{Y}_1^k)] \end{aligned} \tag{23}$$

and

$$\begin{aligned} &\langle \nabla_2 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}), \boldsymbol{U}_2^k\rangle + h(\boldsymbol{U}_2^k) \\ &\leq \langle \nabla_2 f(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}), \boldsymbol{U}_2^*\rangle + h(\boldsymbol{U}_2^*). \end{aligned} \tag{24}$$

Then noting that $D(x,x') \geq 0$ and using the convexity of $f(\cdot)$ and $h(\cdot)$ together with the definition of $\boldsymbol{Z}^k$ in algorithm 1 and eqs. (21), (23) and (24), we end up with:

$$\begin{aligned}
F(\boldsymbol{X}^k,\boldsymbol{Z}^k) \leq & (1-\gamma_k) f(\boldsymbol{X}^{k-1},\boldsymbol{Z}^{k-1}) \\
& + \gamma_k l_f(\boldsymbol{Y}_1^{k-1},\boldsymbol{Y}_2^{k-1};\boldsymbol{U}_1^*,\boldsymbol{U}_2^*) \\
& + \frac{\gamma_k}{2\xi_k}[\boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^{k-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^k)] \\
& + \gamma_k(h(\boldsymbol{U}_2^*) - h(\boldsymbol{U}_2^k)) + h(\boldsymbol{Z}^k) \\
& + \frac{L}{2}\gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_1^k,\boldsymbol{U}_1^{k-1}) + \frac{L}{2}\gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_2^k,\boldsymbol{U}_2^{k-1}) \\
\leq & (1-\gamma_k) f(\boldsymbol{X}^{k-1},\boldsymbol{Z}^{k-1}) + \gamma_k f(\boldsymbol{U}_1^*,\boldsymbol{U}_2^*) \\
& + \frac{L}{2}\gamma_k[\boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^{k-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^k)] \\
& + \gamma_k(h(\boldsymbol{U}_2^*) - h(\boldsymbol{U}_2^k)) + h(\boldsymbol{Z}^k) \\
& + \frac{L}{2}\gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_1^k,\boldsymbol{U}_1^{k-1}) + \frac{L}{2}\gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_2^k,\boldsymbol{U}_2^{k-1}) \\
\leq & (1-\gamma_k) F(\boldsymbol{X}^{k-1},\boldsymbol{Z}^{k-1}) + \gamma_k F(\boldsymbol{U}_1^*,\boldsymbol{U}_2^*) \\
& + \frac{L}{2}\gamma_k[\boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^{k-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^k)] \\
& - \gamma_k h(\boldsymbol{U}_2^k) + h(\boldsymbol{Z}^k) - (1-\gamma_k) h(\boldsymbol{Z}^{k-1}) \\
& + \frac{L}{2}\gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_1^k,\boldsymbol{U}_1^{k-1}) + \frac{L}{2}\gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_2^k,\boldsymbol{U}_2^{k-1}) \\
\leq & (1-\gamma_k) F(\boldsymbol{X}^{k-1},\boldsymbol{Z}^{k-1}) + \gamma_k F(\boldsymbol{U}_1^*,\boldsymbol{U}_2^*) \\
& + \frac{L}{2}\gamma_k[\boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^{k-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^k)] \\
& + \frac{L}{2}\gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_1^k,\boldsymbol{U}_1^{k-1}) + \frac{L}{2}\gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_2^k,\boldsymbol{U}_2^{k-1}). \qquad (25)
\end{aligned}$$

Subtracting $F(\boldsymbol{U}_1^*,\boldsymbol{U}_2^*)$ from both sides of the above inequality, we have:

$$\begin{aligned}
F(\boldsymbol{X}^k,\boldsymbol{Z}^k) - F(\boldsymbol{U}_1^*,\boldsymbol{U}_2^*) \leq & \\
& (1-\gamma_k)(F(\boldsymbol{X}^{k-1},\boldsymbol{Z}^{k-1}) - F(\boldsymbol{U}_1^*,\boldsymbol{U}_2^*)) \\
& + \frac{L}{2}\gamma_k[\boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^{k-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^k)] \\
& + \frac{L}{2}\gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_1^k,\boldsymbol{U}_1^{k-1}) + \frac{L}{2}\gamma_k^2 \boldsymbol{D}(\boldsymbol{U}_2^k,\boldsymbol{U}_2^{k-1}). \qquad (26)
\end{aligned}$$

In view of Lemma 1 of [23] and the definition of γ_k and Γ_k, it is easy to verify that $\frac{\gamma_k^2}{\Gamma_k} = \frac{2k}{k+1} \leq 2$ and $\frac{\gamma_i}{\Gamma_i} = i \leq k$, which implies that:

$$\begin{aligned}
F(\boldsymbol{X}^k,\boldsymbol{Z}^k) - F(\boldsymbol{U}_1^*,\boldsymbol{U}_2^*) \leq & \\
\Gamma_k(1-\gamma_1)(F(\boldsymbol{X}^0,\boldsymbol{Z}^0) - F(\boldsymbol{U}_1^*,\boldsymbol{U}_2^*)) & \\
+ \frac{\Gamma_k L}{2}\sum_{i=1}^k \frac{\gamma_i}{\Gamma_i}[\boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^{i-1}) - \boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^i)] & \\
+ \frac{\Gamma_k L}{2}\sum_{i=1}^k \frac{\gamma_i^2}{\Gamma_i}[\boldsymbol{D}(\boldsymbol{U}_1^i,\boldsymbol{U}_1^{i-1}) + \boldsymbol{D}(\boldsymbol{U}_2^i,\boldsymbol{U}_2^{i-1})]. \qquad (27)
\end{aligned}$$

Let $D_{max} = \max_{x,y\in\Omega}\|x-y\|$ and note that $D(x,x') \geq 0$. We finally have:

$$\begin{aligned}
F(\boldsymbol{X}^k,\boldsymbol{Z}^k) - F^* \leq & \frac{L}{k(k+1)}\{k\boldsymbol{D}(\boldsymbol{U}_1^*,\boldsymbol{Y}_1^0) \\
+ 2\sum_{i=1}^k & [\boldsymbol{D}(\boldsymbol{U}_1^i,\boldsymbol{U}_1^{i-1}) + \boldsymbol{D}(\boldsymbol{U}_2^i,\boldsymbol{U}_2^{i-1})]\} \leq \frac{5LD_{max}^2}{k+1}. \qquad (28)
\end{aligned}$$

Therefore, the algorithm still achieves the optimal rate $O(1/k)$, i.e., a rate of $O(1/\epsilon)$. □

V. The Parallel Algorithm

The above algorithm may be computational expensive when the number of anchor points q increases to an extremely large value. We further speed up the algorithm to accommodate the need of a large number of anchor points q to fit big industry data. To this end, we first rewrite the optimal function in the form of eq. (31). We show in theorem 2 that when λ^τ and β^τ are properly chosen, the two formulations will result in the same optimum. As all the variables $\{\boldsymbol{X}^{s_\tau} = [\boldsymbol{H}^{s_\tau};\ \boldsymbol{A}^{s_\tau}]\}_{\tau=1}^q$ are independent, we develop the parallel method in algorithm 2 that optimizes each block matrix $\{\boldsymbol{X}^{s_\tau}\}_{\tau=1}^q$ separately. Hence, it allows us to deal with the objective function in parallel and makes the algorithm more efficient for big data.

Denote the log-likelihood of observing sequence $\mathcal{T}^{u,i}$ mapping to a specific anchor point $s_\tau = (a_\tau, b_\tau)$ as:

$$\begin{aligned}
\mathcal{L}^{s_\tau}(\mathcal{T}^{u,i} \mid \boldsymbol{X}^{s_\tau}) = & \\
\frac{1}{q}\{\sum_{t_j^{u,i}\in\mathcal{T}^{u,i}} & \log(\boldsymbol{X}(s_\tau)_{u,i}^\top \Phi(s_\tau)_j^{u,i}) - \boldsymbol{X}(s_\tau)_{u,i}^\top \Psi(s_\tau)^{u,i}\}, \qquad (29)
\end{aligned}$$

where:

$$\begin{aligned}
\boldsymbol{X}(s_\tau)_{u,i} =& (\boldsymbol{H}^{s_\tau}(u,i), \boldsymbol{A}^{s_\tau}(u,i))^\top, \\
\Phi(s_\tau)_j^{u,i} =& q(1, \sum_{t_k^{u,i}<t_j^{u,i}} \kappa_\sigma(t_j^{u,i} - t_k^{u,i}))^\top \cdot K_h^{s_\tau}(u,i)/C_{u,i}, \\
\Psi(s_\tau)^{u,i} =& q(T, \sum_{t_j^{u,i}\in\mathcal{T}^{u,i}} \int_{t_j^{u,i}}^T \kappa_\sigma(t - t_j^{u,i})dt)^\top \cdot K_h^{s_\tau}(u,i)/C_{u,i}. \qquad (30)
\end{aligned}$$

Then we define the parallel objective function as:

$$\begin{aligned}
OPT_p = \min_{\boldsymbol{X}^{s_\tau},\boldsymbol{Z}^{s_\tau}} & \frac{1}{|\mathcal{O}|}\sum_{\tau=1}^q\{\sum_{\mathcal{T}^{u,i}\in\mathcal{O}} \mathcal{L}^{s_\tau}(\mathcal{T}^{u,i}|\boldsymbol{X}^{s_\tau}) + h_{s_\tau}(\boldsymbol{Z}^{s_\tau})\} \\
& s.t.\ \boldsymbol{X} \geq \boldsymbol{0}, \qquad (31)
\end{aligned}$$

where $h_{s_\tau}(\boldsymbol{Z}^{s_\tau}) = \lambda^\tau\|\boldsymbol{H}^{s_\tau}\|_* + \beta^\tau\|\boldsymbol{A}^{s_\tau}\|_*$.

Theorem 2. *With the condition that λ^τ and β^τ for $\tau = 1,...,q$ satisfy eq.* (32)*, the optimal value OPT_p in eq.* (31) *coincides with the global optimal value OPT in eq.* (14)*.*

$$\lambda\|\boldsymbol{H}'\|_* + \beta\|\boldsymbol{A}'\|_* \leq \sum_{\tau=1}^q(\lambda^\tau\|\boldsymbol{H}^{s_\tau}\|_* + \beta^\tau\|\boldsymbol{A}^{s_\tau}\|_*). \qquad (32)$$

Proof. For a real convex function $\varphi(\cdot)$, a set of numbers $x_1, x_2, ..., x_n$, and positive weights α_i, Jensen's inequality can be stated as:

$$\varphi(\frac{\sum\alpha_i x_i}{\alpha_i}) \leq \frac{\sum\alpha_i\varphi(x_i)}{\sum\alpha_i}. \qquad (33)$$

The equality holds if and only if $x_1 = x_2 = ... = x_n$ or $\varphi(\cdot)$ is linear. Specifically, eq. (33) becomes:

$$\varphi(\frac{\sum x_i}{n}) \leq \frac{\sum\varphi(x_i)}{n} \qquad (34)$$

if the weighs α_i are equal.

Algorithm 2: Local Low-Rank Hawkes Parallel

Input: All the training events $\mathcal{O} = \{\mathcal{T}^{u,i}\}_{u,i}$; learning rate ξ_k; parameters ρ, λ, β; number of anchor points q; kernel function $K(\cdot)$ of widths h_1, h_2; step size $\gamma_k \in [0, 1]$;

Output: $\{\boldsymbol{X}^{s_\tau} = [\boldsymbol{H}^{s_\tau}; \boldsymbol{A}^{s_\tau}]\}_{\tau=1}^{q}$, which are the set of local parameter matrices:

for $\tau = 1, ..., q$ ***in parallel*** **do**
- $(a_\tau, b_\tau) :=$ a random selected (u, i) pair;
- **for** $i = 1 \to m$ **do**
  - $K_{h_1}^{a_\tau}(i) := \exp(-\frac{1}{2}h^{-2}d(a_\tau, i)^2)$;
- **end**
- **for** $j = 1 \to n$ **do**
  - $K_{h_2}^{b_\tau}(i) := \exp(-\frac{1}{2}h^{-2}d(b_\tau, j)^2)$;
- **end**
- Choose to initialize $\boldsymbol{U}_1^0$;
- Set $\boldsymbol{X}^0 = \boldsymbol{Z}^0 = \boldsymbol{U}_1^0 = \boldsymbol{U}_2^0$;
- **for** $k \leftarrow$ *1 to MaxIter* **do**
  - Set $\boldsymbol{Y}_1^{k-1} = (1 - \gamma_k)\boldsymbol{X}^{k-1} + \gamma_k \boldsymbol{U}_1^{k-1}$;
  - Set $\boldsymbol{Y}_2^{k-1} = (1 - \gamma_k)\boldsymbol{Z}^{k-1} + \gamma_k \boldsymbol{U}_2^{k-1}$;
  - Compute the proximal operator for $\boldsymbol{X}$:
  - $\boldsymbol{U}_1^k = (\boldsymbol{Y}_1^{k-1} - \xi_k \nabla_1 (f_{s_\tau}(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1})))_+$;
  - Use a local linear expansion of f for $\boldsymbol{Z}$:
  - $\boldsymbol{U}_2^k =$
  - $arg\min_{\boldsymbol{Z}} \{\langle \nabla_2 f_{s_\tau}(\boldsymbol{Y}_1^{k-1}, \boldsymbol{Y}_2^{k-1}), \boldsymbol{Z}\rangle + h_{s_\tau}(\boldsymbol{Z})\}$;
  - Set $\boldsymbol{X}^k = (1 - \gamma_k)\boldsymbol{X}^{k-1} + \gamma_k \boldsymbol{U}_1^k$;
  - Set $\boldsymbol{Z}^k = (1 - \gamma_k)\boldsymbol{Z}^{k-1} + \gamma_k \boldsymbol{U}_2^k$;
- **end**

end

As $-log(\cdot)$ is convex, we rewrite eq. (9) based on eq. (34) as:

$$
\begin{aligned}
&- \mathcal{L}(\mathcal{T}^{u,i} \mid \{\boldsymbol{X}^{s_\tau}\}_{\tau=1}^{q}) = \\
&- \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \log(\sum_{\tau=1}^{q} \boldsymbol{X}(s_\tau)_{u,i}^\top \Phi(s_\tau)_j^{u,i}/q) + \sum_{\tau=1}^{q} \boldsymbol{X}(s_\tau)_{u,i}^\top \Psi(s_\tau)^{u,i}/q, \\
&\leq - \sum_{\tau=1}^{q} \{ \sum_{t_j^{u,i} \in \mathcal{T}^{u,i}} \log(\boldsymbol{X}(s_\tau)_{u,i}^\top \Phi(s_\tau)_j^{u,i}) - \boldsymbol{X}(s_\tau)_{u,i}^\top \Psi(s_\tau)^{u,i} \}/q \\
&= - \sum_{\tau=1}^{q} \mathcal{L}^{s_\tau}(\mathcal{T}^{u,i} \mid \boldsymbol{X}^{s_\tau}). \qquad (35)
\end{aligned}
$$

Given eq. (32), we have:

$$
\begin{aligned}
h(\boldsymbol{Z}) &= \lambda \|\boldsymbol{H}'\|_* + \beta \|\boldsymbol{A}'\|_* \\
&\leq \sum_{\tau=1}^{q} (\lambda^\tau \|\boldsymbol{H}^{s_\tau}\|_* + \beta^\tau \|\boldsymbol{A}^{s_\tau}\|_*) = \sum_{\tau=1}^{q} h_{s_\tau}(\boldsymbol{Z}^{s_\tau}). \qquad (36)
\end{aligned}
$$

Therefore, plugging eqs. (35) and (36) into the previous objective function in eq. (11), we have $OPT \leq OPT_p$ and readily arrive at the theorem. □

Therefore, we can optimize the parallel objective function in eq. (31) separately by using the parallel algorithm to approximate the parameter estimation. As the form of the objective function is the same as the global one, we can still use the global updating approach. The details are described in algorithm 2. By assuming q machines for computing, the algorithm can run in parallel to estimate the q local model parameters. In the end, they synchronize to obtain the final results. The parallel algorithm should be q times faster without considering the communication cost.

VI. Experiments

In this section, we present the results of the experiments.

A. Datasets and Evaluation Criteria

We evaluate our model on the three real-world datasets: **IPTV** dataset [11] records the viewing behaviors of 7100 users on 436 TV programs, e.g., what and when they watch, from January to November 2012. It contains 4726 (u, i) pairs with nearly 2.4M events and 1420 movie features such as genres. These features are only used for Coevolve baseline. **Yelp**[1] is available from Yelp dataset challenge. We select users with at least 100 posts, and it contains 35k reviews for 17k businesses by 100 users in 11 years. **Reddit**[2] dataset contains a random selected 1000 users, 1403 groups, and 10k discussions events in January 2014.

We can evaluate the performance of our Hawkes model on two tasks:

Item Relevance: We report Mean Average Rank (MAR) [5] of all the testing items. For a specific user u, we compute the survival $S_{(u,i)}(t) = \exp(-\int_{t_n^{u,i}}^{t} \lambda_{(u,i)}(\tau)d(\tau))$ for all the items at every testing time t. According to the survival, we rank all the items in ascending order, and the real testing item should rank one ideally. Therefore, a smaller value represents better predictive performance.

Time Prediction: We report the Mean Absolute Error (MAE) [3], [5] between the predicted times and the ground truth. We compute the predicted time by calculating the density as $f(t) = \lambda_{(u,i)}(t)S_{(u,i)}(t)$, and then use the expectation to predict the next event. Furthermore, we give the relative percentage of the prediction error (Err %).

B. Baseline Methods and Parameter Settings

Poisson process is a relaxation of Hawkes process with no triggering kernel capturing temporal dependencies. It only contains a base intensity η, which is a constant. The Poisson process is a strong baseline in many cases, as most popular items usually have large base intensities.

PoissonTensor uses Poisson regression other than RMSE as the loss function to fit the number of events, which actually can be considered as the intensity in each discretized time slot [29]. Because the missing values are not random, simulating the values with Poisson distribution is more reasonable than with Gaussian. Once we get the values, there are two ways to simulate the intensity of test data. One is using the intensity that we have got only in the last time interval, and the other

[1] https://www.yelp.com/dataset/challenge

[2] https://dynamics.cs.washington.edu/data.html

is using the average intensity of all the training time intervals. We report the best performance of these two choices.

LowRankHawkes is a Hawkes process based model [3] that can be seen as a relaxation of our model with only one anchor point. It assumes that all the (u, i) pairs are independent so there is no user-item interactions between pairs.

Coevolve is a coevolutionary latent feature process [5] which can be seen as a squared Hawkes process adding a base intensity. It uses user and item features as well as the interaction features between users and items, such as review features, to simulate the intensity of each (u, i) pair. In our experiments, we only use the item feature. If no features are provided, the model reduces to the Poisson process.

Parameter Settings In the experiments, T is the length of the total time, and $p = 0.76$ is the proportion where we split the data. Specifically, we use the events before time $T \cdot p$ as the training data, and the rest of them as testing data. We do experiments on several types of kernels, and find that these do not affect the performance much. We use the Gaussian kernel with $h_1 = h_2 = 0.8$ and report the averaged results on the two tasks above.

C. Results

We show the results of our method using the global algorithm and other baseline methods in Fig. 1, Fig. 2, and Fig. 3 for IPTV, Yelp, and Reddit data, respectively.

Generally, our model outperforms most other baseline methods in item prediction and returning time prediction. The main reason is that each (u, i) pair's intensity is simulated with its own sequence mapping to a total of q local models in our model. **Coevolve** relies on the auxiliary features. **LowRankHawkes** treats each (u, i) pair's process independently. **Poisson** and **PoissonTensor** simulate events without the history, and thus are lack of prediction power.

For IPTV and Reddit data, the exception occurs at the time prediction of **Coevolve**, because the auxiliary feature information is added to this model. The **Coevolve** method uses a weighted summation of all the events happened before the current event to simulate one (u, i) pair's intensity $\lambda^{u,i}(t)$. Therefore, the returning-time prediction is good since a large number of events are used to simulate the intensity function and a huge amount of auxiliary feature information is incorporated. However, the item rank prediction becomes worse [5] because the individual preferences are influenced by the general preference. Meanwhile, we can see that the Hawkes process based models, such as our model, **Coevolve**, and **LowRankHawkes**, get better performances when there are sufficient history events (with nearly 400 events per (u, i) pairs) in comparison with the Poisson related models.

For Yelp data, as each (u, i) pair only has fewer than 3 events in average, the time prediction is similar among **LowRankHawkes** and **Poisson**, which means that the history is not such an important factor. In this time sparsity case, factorization model **PoissonTensor** gets better results than point process based models. Even adding some auxiliary features, the **Coevolve** model achieves comparable performance, our model performs the best without any features. As more information is used to simulate the Hawkes process for each (u, i) pair, our model integrates some interaction influences from similar groups. In other words, our model performs better on sequences without sufficient events.

TABLE I
AVERAGE PERFORMANCE WITH DIFFERENT NUMBERS OF ANCHOR POINTS BY THE GLOBAL ALGORITHMS ON IPTV, YELP, AND REDDIT DATASETS.

Datasets	Metrics	Anchor Points				
		1	2	5	10	15
IPTV	MAR	5.175	2.934	1.705	1.667	1.666
	MAE	822.1	716.3	620.1	449.0	383.0
	Err %	12.67	10.82	9.15	6.44	5.46
Yelp	MAR	116.0	106.6	94.63	95.74	95.09
	MAE	845.7	805.9	520.7	581.7	591.0
	Err %	23.71	22.74	14.98	17.34	17.62
Reddit	MAR	49.14	11.50	6.177	6.129	6.062
	MAE	8476	8117	6909	6138	5508
	Err %	21.50	20.60	17.64	15.88	14.41

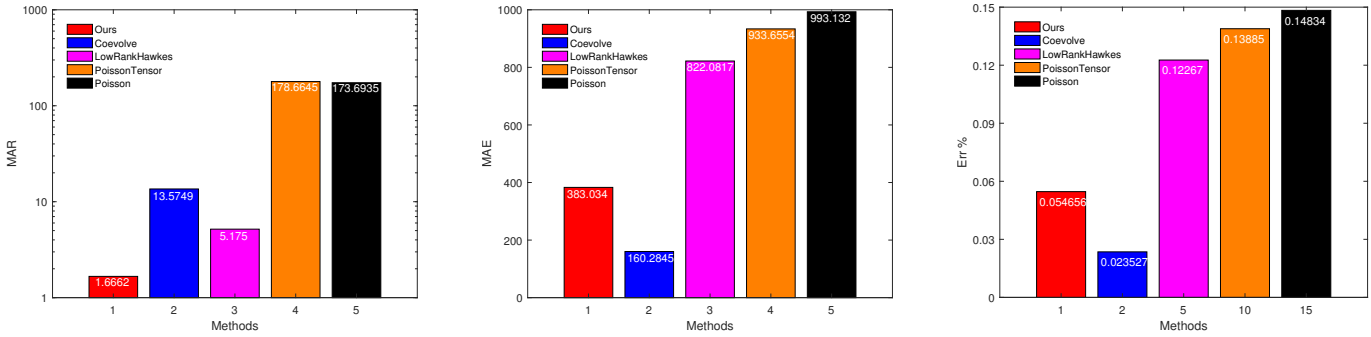
It is worth mentioning that our method achieves good performance on item prediction as it integrates similarity measurements using kernels with the Hawkes process. The kernel measures the similarity between different (u, i) pairs and some anchor points are selected from the user-item dimension. As our model assumes that all the (u, i) pairs can be clustered into several groups, it is more efficient than the multivariate Hawkes process, especially in modeling large-scale interaction events.

1) Effect of Anchor points: We show the results of kernel smoothing with different numbers of anchor points in table I for IPTV, Yelp, and Reddit datasets. The results come from the global algorithm.

For IPTV and Reddit data, both item prediction and returning-time prediction are improved when the number of anchor points increases. The results also indicate the bottleneck of the performance given enough anchor points. As the (u, i) pairs are sparse in the space, i.e., we only have 4726 (u, i) pairs on a 7100×436 matrix for IPTV data, it is not appropriate to set too many anchor points. Therefore, the number of anchor points should depend on the sparsity of the pairs on user-item dimension. It is also worth mentioning that just a few anchor points, e.g., 5, can render pretty good results.

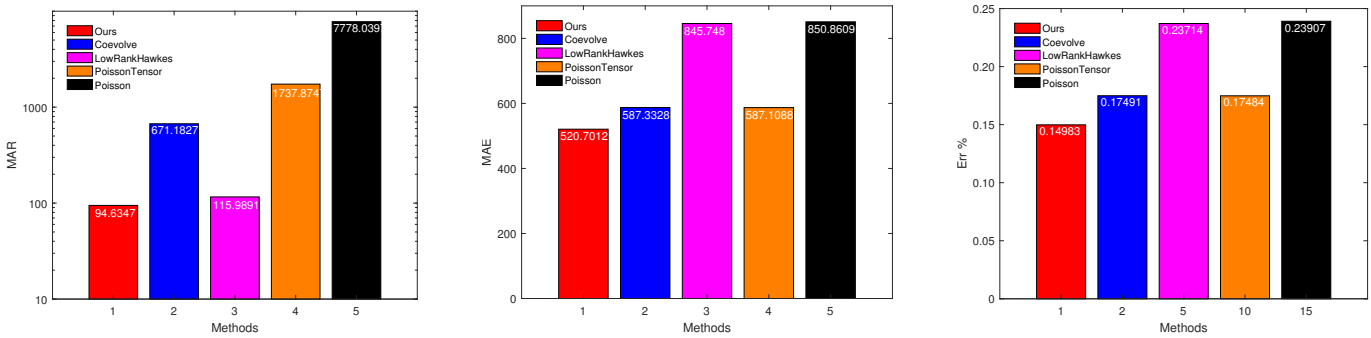
For Yelp data, as the dataset only has 100 users, it reaches the best performance when the number of anchor points is in the range of $(5 \sim 10)$. However, when the number of anchor points further increases, bias may be introduced as some anchor points are similar, so it actually calculates one type of local neighbors repeatedly in eq. (7), which finally lowers the prediction performance. Therefore, selecting anchor points should also depend on the user dimension and the item dimension rather than the pairs' sparsity only.

2) Comparison of Global and Parallel: We also compare the results of the global and parallel algorithms and present



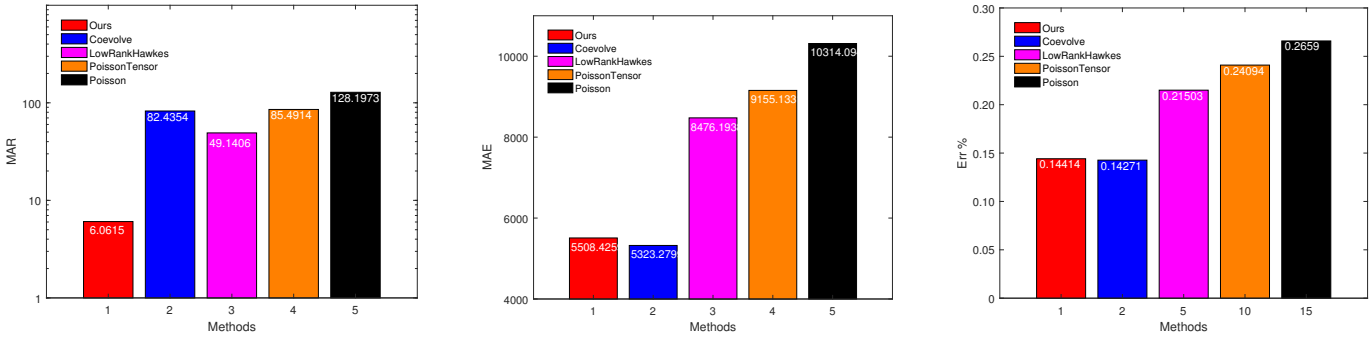
(a) Item relevance (b) Time prediction (c) Err %

Fig. 1. Prediction accuracy on IPTV data



(a) Item relevance (b) Time prediction (c) Err %

Fig. 2. Prediction accuracy on Yelp data



(a) Item relevance (b) Time prediction (c) Err %

Fig. 3. Prediction accuracy on Reddit data

the results in table II. It is obvious that the parallel algorithm performs better than the global algorithm and achieves similar results with a smaller number of anchor points. The reason is that the parallel algorithm is more flexible in controlling the nuclear norm for parameter matrices in comparison with the global algorithm, which only assumes the combination of a number of block matrices low-rank. Specifically, for the global algorithm, only three parameters (λ,β,ρ) are used to control the nuclear norm of model parameter X. For the parallel algorithm, there are up to $3 \cdot q$ parameters in total and each tuple (λ_τ,β_τ,ρ_τ) can be used to control the rank for each local model. Therefore, the parallel algorithm is more compatible with the local low-rank assumption when dealing with the nuclear norm. In the experiments, however, we find that it only slightly improves the results, so we choose the same parameters for all local models with the nuclear norm. Meanwhile, we can see that the prediction accuracy will converge as the number of anchor points grows.

VII. Conclusions

In this paper, we present a novel framework that integrates the kernel smoothing and the Hawkes process to model the temporal events of user-item interactions. We assume that

TABLE II
AVERAGE PERFORMANCE WITH DIFFERENT NUMBERS OF ANCHOR POINTS BY GLOBAL AND PARALLEL ALGORITHMS ON IPTV DATASET.

	Metrics	Anchor Points				
		1	2	5	10	15
Global	MAR	5.175	2.934	1.705	1.667	1.666
	MAE	822.1	716.3	620.1	449.0	383.0
	Err %	12.67	10.82	9.15	6.44	5.46
Parallel	MAR	5.136	2.865	1.684	1.678	1.676
	MAE	822.2	713.7	486.3	379.0	362.7
	Err %	12.33	10.62	7.06	5.40	5.16

the intensity parameter matrix is locally low-rank. With non-parametric kernel smoothing, each user-item pair can be simulated by a series of local matrix mappings. We design an efficient convex optimization algorithm to estimate model parameters and present a parallel algorithm to further increase the computation efficiency. Extensive experiments on real-world datasets demonstrate the performance improvements of our model in comparison with the state of the art. Our model can be applied to other 2D aggregated Hawkes processes, such as temporal user interactions in social networks, and extended to n-dimensional aggregated Hawkes processes, as long as these dimensions satisfy the local low-rank assumption. Further work includes extending to other application areas and integrating the framework with certain deep neural network structures.

VIII. ACKNOWLEDGEMENT

This work was supported in part by the Louisiana Board of Regents under Grant LEQSF(2017-20)-RD-A-29. The authors would also like to thank Yichen Wang and Le Song from Georgia Tech for their helpful discussions.

REFERENCES

[1] K. Zhou, H. Zha, and L. Song, "Learning social infectivity in sparse low-rank networks using multi-dimensional Hawkes processes," in *Proc. of the 16th International Conference on Artificial Intelligence and Statistics*, 2013, pp. 641–649.

[2] M. Farajtabar, N. Du, M. G. Rodriguez, I. Valera, H. Zha, and L. Song, "Shaping social activity by incentivizing users," in *Proc. of the 28th Annual Conference on Advances in Neural Information Processing Systems*, 2014, pp. 2474–2482.

[3] N. Du, Y. Wang, N. He, J. Sun, and L. Song, "Time-sensitive recommendation from recurrent user activities," in *Proc. of the 29th Annual Conference on Advances in Neural Information Processing Systems*, 2015, pp. 3492–3500.

[4] W. Xiao, X. Xu, K. Liang, J. Mao, and J. Wang, "Job recommendation with Hawkes process: an effective solution for RecSys Challenge 2016," in *Proc. of the Recommender Systems Challenge*, 2016.

[5] Y. Wang, N. Du, R. Trivedi, and L. Song, "Coevolutionary latent feature processes for continuous-time user-item interactions," in *Proc. of the 30th Annual Conference on Advances in Neural Information Processing Systems*, 2016, pp. 4547–4555.

[6] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.

[7] T. J. Liniger, "Multivariate Hawkes processes," Ph.D. dissertation, ETH Zurich, 2009.

[8] R. Lemonnier, K. Scaman, and A. Kalogeratos, "Multivariate Hawkes processes for large-scale inference." in *Proc. of the AAAI Conference on Artificial Intelligence*, 2017, pp. 2168–2174.

[9] J. Etesami, N. Kiyavash, K. Zhang, and K. Singhal, "Learning network of multivariate Hawkes processes: a time series approach," in *Proc. of the 32nd Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2016, pp. 162–171.

[10] M. Eichler, R. Dahlhaus, and J. Dueck, "Graphical modeling for multivariate Hawkes processes with nonparametric link functions," *Journal of Time Series Analysis*, vol. 38, no. 2, pp. 225–242, 2017.

[11] H. Xu, M. Farajtabar, and H. Zha, "Learning granger causality for Hawkes processes," in *Proc. of the 33rd International Conference on Machine Learning*, 2016, pp. 1717–1726.

[12] M. Krumin, I. Reutsky, and S. Shoham, "Correlation-based analysis and generation of multiple spike trains using Hawkes models with an exogenous input," *Frontiers in Computational Neuroscience*, vol. 4, p. 147, 2010.

[13] E. C. Hall and R. M. Willett, "Tracking dynamic point processes on networks," *IEEE Transactions on Information Theory*, vol. 62, no. 7, pp. 4327–4346, 2016.

[14] J. Lee, S. Kim, G. Lebanon, and Y. Singer, "Local low-rank matrix approximation." in *Proc. of the International Conference on Machine Learning (ICML)*, vol. 28, 2013, pp. 82–90.

[15] J. Lee, M. Sun, S. Kim, and G. Lebanon, "Automatic feature induction for stagewise collaborative filtering," in *Proc. of Annual Conference on Neural Information Processing Systems (NIPS)*, Dec. 2012, pp. 314–322.

[16] J. Lee, M. Sun, and G. Lebanon, "PREA: Personalized recommendation algorithms toolkit," *Journal of Machine Learning Research (JMLR)*, vol. 13, no. 1, pp. 2699–2703, Sept. 2012.

[17] M. Sun, G. Lebanon, and P. Kidwell, "Estimating probabilities in recommendation systems," in *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, Apr. 2011, pp. 734–742.

[18] H. Xu, W. Wu, S. Nemati, and H. Zha, "Patient flow prediction via discriminative learning of mutually-correcting processes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 157–171, 2017.

[19] D. R. Cox and P. A. W. Lewis, "Multivariate point processes," in *Proc. 6th Berkeley Symp. Math. Statist. Prob*, vol. 3, 1972, pp. 401–448.

[20] O. Aalen, O. Borgan, and H. Gjessing, *Survival and Event History Analysis: A Process Point of View*. Springer Science & Business Media, 2008.

[21] M. P. Wand and M. C. Jones, *Kernel Smoothing*. Chapman and Hall/CRC, 1995.

[22] S. Sastry, "Some NP-complete problems in linear algebra," *Honors Projects*, 1990.

[23] G. Lan, "The complexity of large-scale convex programming under a linear optimization oracle," *arXiv preprint arXiv:1309.5550*, 2013.

[24] ——, "An optimal method for stochastic composite optimization," *Mathematical Programming*, vol. 133, no. 1, pp. 365–397, 2012.

[25] Y. Nesterov, "Gradient methods for minimizing composite functions," *Mathematical Programming*, vol. 140, no. 1, pp. 125–161, 2013.

[26] H. Dai, Y. Wang, R. Trivedi, and L. Song, "Recurrent coevolutionary latent feature processes for continuous-time recommendation," in *Proc. of the 1st Workshop on Deep Learning for Recommender Systems*, 2016, pp. 29–34.

[27] R. Bell, Y. Koren, and C. Volinsky, "Modeling relationships at multiple scales to improve accuracy of large recommender systems," in *Proc. of the ACM SIGKDD*, 2007.

[28] A. W. Yu, W. Ma, Y. Yu, J. Carbonell, and S. Sra, "Efficient structured matrix rank minimization," in *Proc. of the 28th Annual Conference on Advances in Neural Information Processing Systems*, 2014, pp. 1350–1358.

[29] E. C. Chi and T. G. Kolda, "On tensors, sparsity, and nonnegative factorizations," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1272–1299, 2012.