FASTEN: Fast Sylvester Equation Solver for Graph Mining

Boxin Du
Arizona State University
boxin.du@asu.edu

Hanghang Tong
Arizona State University
hanghang.tong@asu.edu

ABSTRACT

The Sylvester equation offers a powerful and unifying primitive for a variety of important graph mining tasks, including network alignment, graph kernel, node similarity, subgraph matching, etc. A major bottleneck of Sylvester equation lies in its high computational complexity. Despite tremendous effort, state-of-the-art methods still require a complexity that is at least *quadratic* in the number of nodes of graphs, even with approximations. In this paper, we propose a family of Krylov subspace based algorithms (FASTEN) to speed up and scale up the computation of Sylvester equation for graph mining. The key idea of the proposed methods is to project the original equivalent linear system onto a Kronecker Krylov subspace. We further exploit (1) the implicit representation of the solution matrix as well as the associated computation, and (2) the decomposition of the original Sylvester equation into a set of inter-correlated Sylvester equations of smaller size. The proposed algorithms bear two distinctive features. First, they provide the *exact* solutions without any approximation error. Second, they significantly reduce the time and space complexity for solving Sylvester equation, with two of the proposed algorithms having a *linear* complexity in both time and space. Experimental evaluations on a diverse set of real networks, demonstrate that our methods (1) are up to $10,000\times$ faster against Conjugate Gradient method, the best known competitor that outputs the exact solution, and (2) scale up to million-node graphs.

1 INTRODUCTION

How can we link users from different social network sites (e.g., Facebook, Twitter, LinkedIn, etc.)? How can we integrate different tissue-specific protein-protein interaction (PPI) networks together to prioritize candidate genes? How to predict the toxicity of chemical molecules by comparing their three-dimensional structure? How can we detect suspicious transaction patterns (e.g., money-laundering ring) in the financial network given a template pattern/query? The Sylvester equation [13], defined over the adjacency matrices of the input networks, provides a powerful and unifying primitive for a variety of key graph mining tasks, including network alignment [22], graph kernel [19], node similarity [11], subgraph matching [4], etc. Figure 1 presents an illustrative example of using Sylvester equation for graph mining. Please refer to Section 2 for its formal definition.

KDD '18, London, United Kingdom

DOI: 10.1145/3219819.3220002

A major limitation of Sylvester equation lies in the high computational complexity. For graphs with n nodes and m edges, the straight-forward solver is $O(n^6)$ in time and $O(m^2)$ in space. Much effort has been devoted to speed up the computation for solving Sylvester equation. Nonetheless, state-of-the-art methods for plain graphs require a complexity that is at least $O(mn + n^2)$ [20]. The complexity for solving Sylvester equation would be even more intensified when the input graphs have accompanying node attributes. For example, it would add an additional $O(l)$ (l is the number of node attribute values) to the time complexity of conjugate gradient descent solver. Some recent approximate algorithms try to reduce this complexity by matrix low-rank approximation. Nonetheless, it still requires $O(n^2)$, even with such an approximation. Table 1 summarizes the time and space complexity of the existing methods for solving Sylvester equations. Consequently, most of the existing Sylvester equation solvers can only handle graphs with up to tens of thousands of nodes.

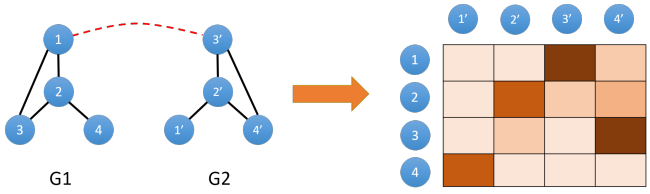


Figure 1: An illustrative example of network alignment by Sylvester equation [9, 20]. Left: the input plain graphs with the dashed line indicating the preference matrix B. Right: the solution matrix X of the corresponding Sylvester equation $X = A_1XA_2 + B$ gives the cross-network node similarity, where A_1 and A_2 are the adjacency matrices of the input graphs; and a darker square on the right represents higher cross-network node similarity. Best viewed in color. See the details in Section 2.

To address this issue, we propose a family of Krylov subspace based algorithms (FASTEN) to speed up and scale up the computation of Sylvester equation for graph mining. The key idea of the proposed methods is to project the original corresponding linear system onto a Kronecker Krylov subspace. By doing so, we are able to obtain an $O(n)$ factor reduction in time complexity and $O(m^2/n^2)$ factor reduction in space complexity compared to normal Krylov subspace based Sylvester equation solver for plain graphs [13]. Building upon that, we seek to further reduce the complexity. Here, our key observation is that the preference matrix in the Sylvester equation is often low-rank (e.g. Figure 1), which implies that the solution matrix itself must have low-rank structure. This, together with some additional optimization (e.g., exploiting the linearity by using the block-diagonal structure of the solution matrix for the attributed graphs), helps further reduces the complexity of the proposed method to be *linear* in both time and space, without

approximation error. Table 1 (shaded parts) summarizes the time and space complexities of the proposed FASTEN algorithms.

The main contributions of the paper are:

- **Novel Algorithms**. We propose a family of efficient and accurate Sylvester equation solver for graph mining tasks, with and without node attribute.
- **Proof and Analysis**. We provide theoretic analysis of the proposed algorithms in terms of the accuracy and complexity.
- **Empirical Evaluations**. We perform extensive experimental evaluations on a diverse set of real networks with a variety of graph mining tasks, which demonstrate that our methods (1) are up to $10,000\times$ faster than the Conjugate Gradient method, the best known competitor that output exact solution , and (2) scale up to million-node graphs.

The rest of the paper is organized as follows. Section 2 formulate the problem of the Sylvester equation for graph mining. Section 3 and 4 present our proposed algorithms in two scenarios, with or without attributes. Section 5 presents the experimental results. In section 6 we review the related work and we conclude the paper in section 7.

Table 1: Complexity Summary and Comparison. r (the rank of input graphs), l (number of node attributes) and k (subspace size) are much smaller compared with m and n. Some small constants are omitted for clarity. See Section 3 and Section 4 for details.)

Algorithm	Attributed(Y/N)	Exact(Y/N)	Time	Space
FP [18, 19]	Y	Y	$O(n^3)$	$O(m^2)$
CG [18, 19]	Y	Y	$O(n^3)$	$O(m^2)$
Sylv. [18, 19]	Y	Y	$O(n^3)$	$O(m^2)$
ARK [7]	Y	N	$O(n^2)$	$O(n^2)$
Cheetah [10]	Y	N	$O(rn^2)$	$O(n^2)$
NI-Sim [9]	N	N	$O(n^2)$	$O(r^2n^2)$
FINAL-P [20]	N	Y	$O(mn + n^2)$	$O(n^2)$
FINAL-NE [20]	Y	Y	$O(lmn + ln^2)$	$O(n^2)$
FINAL-N+ [20]	Y	N	$O(n^2)$	$O(n^2)$
FASTEN-P	N	Y	$O(kn^2)$	$O(n^2)$
FASTEN-P+	N	Y	$O(km + kn)$	$O(m + kn)$
FASTEN-N	Y	Y	$O(mn/l + kn^2/l)$	$O(m/l + n^2)$
FASTEN-N+	Y	Y	$O(km + k^2ln)$	$O(m + kln)$

2 PROBLEM DEFINITION

The main symbols and notations used in the paper are summarized in Table 2. The calligraphic letters $\mathcal{G}_1$ and $\mathcal{G}_2$ represent two attributed graphs. The uppercase bold letters $\mathbf{A_1}$ and $\mathbf{A_2}$ represent the $n \times n$ adjacency matrices and the uppercase bold letters $\mathbf{N_1}$ and $\mathbf{N_2}$ represent the node attribute matrices. $\mathbf{N_1}$ and $\mathbf{N_2}$ are diagonal matrices in which $\mathbf{N}_1^j(a, a) = 1$ if the node a in graph $\mathcal{G}_1$ has node attribute j and otherwise it is zero. The uppercase bold letter $\mathbf{N}$ without subscript or superscript is the combined node attribute matrix of two input graphs. $\mathbf{N} = \prod_{j=1}^{l} \mathbf{N}_1^j \otimes \mathbf{N}_2^j$ where l is the number of node attributes. We use uppercase bold letter $\mathbf{D}$ as the combined diagonal degree matrix of the two input graphs. For a matrix (e.g., $\mathbf{X}$), we vectorize it in the column order to obtain its equivalent vector representation (e.g., $\mathbf{x} = \text{vec}(\mathbf{X})$). For an attributed network, we index its nodes by the corresponding attributes in the adjacency matrix, i.e., all the nodes with the same attribute are consecutive rows/columns in the adjacency matrix, and their induced subgraph is a block of the adjacency matrix with consecutive indices. For example, for $\mathcal{G}_1$ in Figure 2, node-1 and node-2 are the first two rows/columns in the adjacency matrix $\mathbf{A}_1$ since they share the same node attribute (blue diamond); and node-3 and node-4 are the last two rows/columns in $\mathbf{A}_1$ since they share the same node attribute (green hexagon). This will simplify the description of the proposed algorithms.

Table 2: Symbols and Definition

Symbols	Definition
$\mathcal{G}_1 = \{\mathbf{A_1}, \mathbf{N_1}\}$	an attributed graph
$\mathbf{N}$	combined node attribute matrix of input graphs
$\mathbf{R}$, $\mathbf{r}$	residual matrix and residual vector
$\mathbf{H_1}$, $\mathbf{H_2}$	$k \times k$ Hessenberg matrices
$\mathbf{D_1}, \mathbf{D_2}$	diagonal degree matrices
$\mathbf{I}$	an identity matrix
$\mathbf{B}$	preference matrix in the Sylvester equation
$\mathcal{K}_k(\mathbf{A}, \mathbf{r_0})$	Krylov subspace with dimension k
k	Krylov subspace size $k \ll n$
α	the parameter $0 < \alpha < 1$
l	the number of node attribute
$\mathbf{b} = \text{vec}(\mathbf{B})$	vectorize a matrix $\mathbf{B}$ in the column order
$[\mathbf{A}, \mathbf{B}]$	concatenate two matrices in a row
$[\mathbf{A}; \mathbf{B}]$	concatenate two matrices in a column
$\text{diag}(\mathbf{A}_1, ..., \mathbf{A}_i)$	diagonalize i matrices
$\otimes$	Kronecker product
$\text{trace}(\cdot)$	trace of a matrix
$\|\cdot\|_F$	Frobenius norm

Sylvester equation for graph mining. For the completeness, we present a brief review of Sylvester equations and their applications to graph mining. For more details, please refer to [9, 19, 20]. For graphs without attributes, let $\mathbf{A}_1 \leftarrow \alpha^{1/2}\mathbf{D}_1^{-1/2}\mathbf{A}_1\mathbf{D}_1^{-1/2}$ and $\mathbf{A}_2 \leftarrow \alpha^{1/2}\mathbf{D}_2^{-1/2}\mathbf{A}_2\mathbf{D}_2^{-1/2}$ be the normalized adjacency matrices of two input graphs, and $\mathbf{B}$ be the preference matrix. We have the following Sylvester equation [20].

$$\mathbf{X} - \mathbf{A}_2\mathbf{X}\mathbf{A}_1^T = \mathbf{B} \tag{1}$$

By the Kronecker product property, we have the equivalent linear system of Equation (1) as follows.

$$(\mathbf{I} - \mathbf{W})\mathbf{x} = \mathbf{b} \tag{2}$$

where $\mathbf{W} = \mathbf{A}_1 \otimes \mathbf{A}_2$, $\mathbf{b} = \text{vec}(\mathbf{B})$ and $\mathbf{x} = \text{vec}(\mathbf{X})$. We assume that $\mathbf{A}_1$ and $\mathbf{A}_2$ are of the same size $n \times n$. For example, for the two input graphs in Figure 1, $\mathbf{A}_1$ and $\mathbf{A}_2$ are 4×4 adjacency matrices of $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively, and $\mathbf{B}$ is the preference matrix to reflect the prior knowledge (the dashed line). The entries of the solution matrix $\mathbf{X}$ indicate the similarity between a node pair across two input graphs.

When the input graphs have attributes on nodes, the $\mathbf{W}$ matrix in Equation (2) becomes $\mathbf{W} = \mathbf{D}^{-1/2}[\mathbf{N}(\mathbf{A}_1 \otimes \mathbf{A}_2)\mathbf{N}]\mathbf{D}^{-1/2}$. To simplify the notation, let $\mathbf{A}_1^{(ij)} \leftarrow \alpha^{1/2}\mathbf{D}_1^{-1/2}\mathbf{N}_1^i\mathbf{A}_1\mathbf{N}_1^j\mathbf{D}_1^{-1/2}$, $\mathbf{A}_2^{(ij)} \leftarrow \alpha^{1/2}\mathbf{D}_2^{-1/2}\mathbf{N}_2^i\mathbf{A}_2\mathbf{N}_2^j\mathbf{D}_2^{-1/2}$ be the attributed, normalized adjacency matrices. We can see that $\mathbf{A}_1^{(ij)}$ is of the same size of $\mathbf{A}_1$,

but it is 'filtered' by the corresponding attributes. In other words, $\mathbf{A}_1^{(ij)}$ only contains links in $\mathbf{A}_1$ from nodes with attribute i to nodes with attributes j. In this case, Equation (2) becomes:

$$[\mathbf{I} - \sum_{i=1}^{\tilde{\mathcal{O}}} \sum_{j=1}^{\tilde{\mathcal{O}}} (\mathbf{A}_1^{(ij)} \otimes \mathbf{A}_2^{(ij)})]\mathbf{x} = \mathbf{b} \tag{3}$$

Again, by the Kronecker product property, we have the following equivalent Sylvester equation of Equation (3):

$$\mathbf{X} - \sum_{i=1}^{\tilde{\mathcal{O}}} \sum_{j=1}^{\tilde{\mathcal{O}}} \mathbf{A}_2^{(ij)} \mathbf{X} (\mathbf{A}_1^{(ij)})^T = \mathbf{B} \tag{4}$$

For the ease of description of the proposed algorithms, we also use a block-matrix representation. Take $\mathbf{A}_1$ for an example, we have $\mathbf{A}_1 = [\mathbf{A}_1^{ij}]_{i;j=1;:::;l}$, where $\mathbf{A}_1^{ij}$ is a block of matrix $\mathbf{A}_1$ from rows of attribute i to columns of attribute j. Note the subtle difference between $\mathbf{A}_1^{ij}$ and $\mathbf{A}_1^{(ij)}$: they have different sizes and we can verify that $\mathbf{A}_1 = \sum_{i;j=1}^{l} \mathbf{A}_1^{(ij)}$. We use a similar representation for other matrices in Equation (4), i.e., $\mathbf{A}_2 = [\mathbf{A}_2^{ij}]_{i;j=1;:::;l}$, $\mathbf{B} = [\mathbf{B}^{ij}]_{i;j=1;:::;l}$, and $\mathbf{X} = [\mathbf{X}^{ij}]_{i;j=1;:::;l}$.

Figure 2 presents an illustrative example of attributed Sylvester equation. $\mathbf{A}_1$ and $\mathbf{A}_2$ are two 4×4 adjacency matrices, and $\mathbf{N}_1$ and $\mathbf{N}_2$ are the node attribute matrices of $\mathcal{G}_1$ and $\mathcal{G}_2$. $\mathbf{B}$ represents the prior knowledge (the dashed line). Each entry of the solution $\mathbf{X}$ indicates a similarity score between a node pair across $\mathcal{G}_1$ and $\mathcal{G}_2$. Although both $\mathbf{A}_1^{(11)}$ and $\mathbf{A}_1^{11}$ correspond to links between node-*1* and node-*2* (since both nodes share the first attribute, i.e., blue diamond), their sizes are different: $\mathbf{A}_1^{(11)}$ is of 4×4 whereas $\mathbf{A}_1^{11}$ is of 2×2.

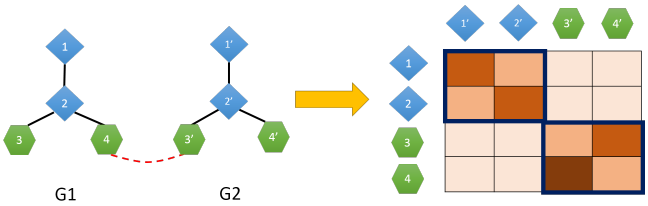


Figure 2: An illustrative example of attributed network alignment by Sylvester equation [9, 20]. Left: the input graphs with node attributes (colors and shapes). Dashed line: preference matrix. Right: the solution matrix X of Equation (4), representing cross-network node similarity. A darker square represents a higher similarity value. Best viewed in color.

The Sylvester equations defined in Equation (1) and Equation (4), together with their equivalent linear systems (Equation (2) and Equation (3)) provide a very powerful tool for many graph mining tasks. For example, the solution matrix $\mathbf{X}$ indicates the cross-network node similarity, which can be directly used for the task of network alignment; by aggregating the solution matrix $\mathbf{X}$ (e.g., by a weighted linear summation over the entries in $\mathbf{X}$), it measures the similarity between the two input graphs [7]; if one of the two input graph is a small query graph, the solution matrix $\mathbf{X}$ becomes the basis for (interactive) subgraph matching [4]; if the two input graphs are identical without node attributes, the corresponding Sylvester equation degenerates to *SimRank* and thus its solution matrix $\mathbf{X}$ measures the node similarity [9].

However, as mentioned earlier, a major bottleneck lies in the high computational complexity. In the next two sections, we present our solutions to speed up and scale up the computation of Sylvester equations, which are divided into two parts based on whether or not the input graphs are attributed. See Table 1 for a summary and comparison.

Krylov subspace methods for linear systems. A classic method for solving a linear system $\mathbf{Ax} = \mathbf{b}$ is via Krylov subspace methods [13]. It first generates an orthogonal basis of the its Krylov subspace with an initial residual vector $\mathbf{r}_0$, denoted by $\mathcal{K}_k(\mathbf{A}, \mathbf{r}_0)$, where $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$, and $\mathbf{x}_0$ is an initial solution. Then it iteratively updates the residual vector and the corresponding solution vector $\mathbf{x}$ over the Krylov subspace formed equation [13] Compared with the alternative methods, e.g., (conjugate) gradient descent, the major advantage of Krylov method lies in the ability to project the original system onto a subspace with a much smaller dimension/size, which can be in turn solved very efficiently.

3 FAST ALGORITHMS FOR PLAIN GRAPHS

In this section, we address the Sylvester equation for plain graphs without node attributes, i.e., Equation (1) and Equation (2). We start with the key ideas and intuition behind the proposed algorithms, and then present the detailed algorithms (FASTEN-P and FASTEN-P+), followed by some analysis in terms of the accuracy and complexity.

3.1 Intuition and Key Ideas

Let us first highlight the key ideas and intuition behind the two proposed algorithms, using the example in Figure 1. First, if we directly apply the standard Krylov subspace methods to solve Equation (2), we would need to generate the orthonormal basis of $\mathcal{K}_{k^2}(\mathbf{I} - \mathbf{A}_1 \otimes \mathbf{A}_2, \mathbf{r}_0)$ or $\mathcal{K}_{k^2}(\mathbf{A}_1 \otimes \mathbf{A}_2, \mathbf{r}_0)$, which requires a complexity as high as $O(n^4)$ in both time and space. To avoid such a high polynomial complexity, our first key idea is to represent the Krylov subspace of $\mathbf{I} - \mathbf{A}_1 \otimes \mathbf{A}_2$ as well as conduct the subsequently computation to update the residual/solution vectors *indirectly* by the Krylov spaces of the two input graphs. Taking Figure 1 for an example, where the two adjacency matrices are both 4×4. The Krylov subspace of the corresponding Sylvester equation is in R^{16}. As will shown in the proposed FASTEN-P algorithm, we will decompose it into the Kronecker product of two Krylov subspace in R^4, which will largely reduce the time and space cost (FASTEN-P).

Second, notice that the solution matrix $\mathbf{X}$ itself is of size $n \times n$. Thus, if we compute and store it in a straight-forward way, it still needs $O(n^2)$ complexity. Here, the key observation is that the preference matrix $\mathbf{B}$ often has a low-rank structure. For the example in Figure 1, the preference matrix $\mathbf{B}$ only has two non-zero entries ($\mathbf{B}(3,4)$ and $\mathbf{B}(4,3)$), making itself a rank-2 matrix. This observation is crucial as it allows us to perform all the intermediate computation as well as to represent the solution matrix $\mathbf{X}$ in an indirect way, leading to a *linear* complexity (FASTEN-P+).

3.2 Proposed FASTEN-P Algorithm

Let $\mathbf{D_1}$ and $\mathbf{D_2}$ be the diagonal degree matrices of the adjacency matrices of the two input graphs $\mathbf{A_1}$ and $\mathbf{A_2}$ respectively. We normalize the adjacency matrices as $\mathbf{A_1} \leftarrow \alpha^{1/2}\mathbf{D_1}^{-1/2}\mathbf{A_1}\mathbf{D_1}^{-1/2}$, and $\mathbf{A_2} \leftarrow \alpha^{1/2}\mathbf{D_2}^{-1/2}\mathbf{A_2}\mathbf{D_2}^{-1/2}$, where $0 < \alpha < 1$ is a regularization parameter.

We first show how to represent $\mathcal{K}_{k^2}(\mathbf{A_1} \otimes \mathbf{A_2}, \mathbf{r}_0)$ *indirectly* in the form $\mathcal{K}_k(\mathbf{A_1}, \mathbf{g}) \otimes \mathcal{K}_k(\mathbf{A_2}, \mathbf{f})$. To be specific, let $\mathbf{V}_i = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_i]$, $\mathbf{W}_j = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_j]$ for $i \in \{k, k+1\}$, and $j \in \{k, k+1\}$, where $\{\mathbf{v}_i\}_{i=1}^{k}$ and $\{\mathbf{w}_j\}_{j=1}^{k}$ are orthonormal basis of $\mathcal{K}_k(\mathbf{A_1}, \mathbf{g})$ and $\mathcal{K}_k(\mathbf{A_2}, \mathbf{f})$, respectively. Let $\mathbf{H_1}$, $\mathbf{H_2}$ and $\tilde{\mathbf{H}}_\mathbf{1}$, $\tilde{\mathbf{H}}_\mathbf{2}$ be the Hessenberg and Hessenberg-like matrices generated by the Arnoldi process [1] [13]. We have that

$$\mathbf{H_1} = \mathbf{V}_k^T\mathbf{A_1}\mathbf{V}_k, \mathbf{H_2} = \mathbf{W}_k^T\mathbf{A_2}\mathbf{W}_k \tag{5}$$

$$\tilde{\mathbf{H}}_1 = \mathbf{V}_{k+1}^T\mathbf{A_1}\mathbf{V}_k, \tilde{\mathbf{H}}_2 = \mathbf{W}_{k+1}^T\mathbf{A_2}\mathbf{W}_k$$

$\tilde{\mathbf{H}}_1, \tilde{\mathbf{H}}_2$ are $k \times k$ and $\tilde{\mathbf{H}}_1, \tilde{\mathbf{H}}_2$ are $(k+1) \times k$. Notice that k is often much smaller than n (i.e., $k \ll n$). Therefore the size of the above Hessenberg and Hessenberg-like matrices is small. We can prove that $\mathbf{V}_k \otimes \mathbf{W}_k$ forms the orthonormal basis of $\mathcal{K}_k(\mathbf{A_1}, \mathbf{g}) \otimes \mathcal{K}_k(\mathbf{A_1}, \mathbf{f})$ (see the details in the proof of Theorem 3.1).

Next, we show how to update the residual vector $\mathbf{r}$ and the solution vector $\mathbf{x}$ using the indirect representation of $\mathcal{K}_{k^2}(\mathbf{A_1} \otimes \mathbf{A_2}, \mathbf{r}_0)$. Let $\mathbf{x}_0$ be an initial solution of Equation (2). The initial residual vector is:

$$\mathbf{r}_0 = \mathbf{b} - (\mathbf{I} - \alpha\mathbf{W})\mathbf{x}_0 \tag{6}$$

To obtain a new solution $\mathbf{x}_1 = \mathbf{x}_0 + \mathbf{z}_0$, we want $\mathbf{z}_0 \in \mathcal{K}_k(\mathbf{A_1}, \mathbf{g}) \otimes \mathcal{K}_k(\mathbf{A_2}, \mathbf{f})$, i.e.,

$$\mathbf{z}_0 = (\mathbf{V}_k \otimes \mathbf{W}_k)\mathbf{y} \tag{7}$$

for an unknown vector $\mathbf{y} \in \mathsf{R}^{k^2}$. The new residual is:

$$\mathbf{r}_1 = \mathbf{r}_0 - [(\mathbf{A}_1 \otimes \mathbf{A}_2)(\mathbf{V}_k \otimes \mathbf{W}_k)\mathbf{y} - (\mathbf{V}_k \otimes \mathbf{W}_k)\mathbf{y}] \tag{8}$$

Based on Equation (5) , we seek to minimize the residual:

$$\begin{aligned}
||\mathbf{r}_1||_2 &= \min_{\mathbf{y}\in\mathsf{R}^{k^2}} ||\mathbf{r}_0 - (\mathbf{A}_1 \otimes \mathbf{A}_2)(\mathbf{V}_k \otimes \mathbf{W}_k)y + (\mathbf{V}_k \otimes \mathbf{W}_k)\mathbf{y}||_2 \\
&= \min_{\mathbf{y}\in\mathsf{R}^{k^2}} ||\mathbf{r}_0 - (\mathbf{A_1}\mathbf{V}_n) \otimes (\mathbf{A_2}\mathbf{W}_k)\mathbf{y} + (\mathbf{V}_k \otimes \mathbf{W}_k)\mathbf{y}||_2 \\
&= \min_{\mathbf{y}\in\mathsf{R}^{k^2}} ||\mathbf{r}_0 - (\mathbf{V}_{k+1}\tilde{\mathbf{H}}_1) \otimes (\mathbf{W}_{k+1}\tilde{\mathbf{H}}_2)\mathbf{y} + (\mathbf{V}_k \otimes \mathbf{W}_k)\mathbf{y}||_2 \\
&= \min_{\mathbf{y}\in\mathsf{R}^{k^2}} ||(\mathbf{V}_{k+1} \otimes \mathbf{W}_{k+1})[(\mathbf{V}_{k+1} \otimes \mathbf{W}_{k+1})^T\mathbf{r}_0 - (\tilde{\mathbf{H}}_1 \otimes \tilde{\mathbf{H}}_2)\mathbf{y} \\
&\quad + (\mathbf{I}_{k+1;k} \otimes \mathbf{I}_{k+1;k})\mathbf{y}]||_2 \\
&= \min_{\mathbf{y}\in\mathsf{R}^{k^2}} ||(\mathbf{V}_{k+1} \otimes \mathbf{W}_{k+1})^T\mathbf{r}_0 - (\tilde{\mathbf{H}}_1 \otimes \tilde{\mathbf{H}}_2)\mathbf{y} + (\mathbf{I}_{k+1;k} \otimes \\
&\quad \mathbf{I}_{k+1;k})\mathbf{y}||_2
\end{aligned} \tag{9}$$

where $\mathbf{I}_{k;k+1} = [\delta_{i;j}]_{1\le i\le k+1;1\le j\le k}$ and $\delta_{i;j}$ is the Kronecker δ-function. Equation (9) can be solved by:

$$\min_{\mathbf{Y}\in\mathsf{R}^{k^2}} ||\mathbf{W}_{k+1}^T\mathbf{R}_0\mathbf{V}_{k+1} - \tilde{\mathbf{H}}_2\mathbf{Y}\tilde{\mathbf{H}}_1^T + \mathbf{I}_{k+1;k}\mathbf{Y}\mathbf{I}_{k+1;k}^T||_F \tag{10}$$

[1]Note that if $\mathbf{A}_1$ and $\mathbf{A}_2$ are symmetric (undirected graphs), Lanczos algorithm can be applied. We use Arnoldi process for generality.

It can be proved that the least square problem in equation (10) can be solved by the following linear system (see details in the proof of Theorem 3.1):

$$\mathscr{L}(\mathbf{Y}) = \mathbf{C} \tag{11}$$

where $\mathscr{L}(\mathbf{Y}) = \tilde{\mathbf{H}}_2^T\tilde{\mathbf{H}}_2\mathbf{Y}\tilde{\mathbf{H}}_1^T\tilde{\mathbf{H}}_1 - \mathbf{H}_2^T\mathbf{Y}\mathbf{H}_1 - \mathbf{H}_2\mathbf{Y}\mathbf{H}_1^T + \mathbf{Y}$, and $\mathbf{C} = \tilde{\mathbf{H}}_2^T\mathbf{W}_{k+1}^T\mathbf{R}_0\mathbf{V}_{k+1}\tilde{\mathbf{H}}_1 - \mathbf{W}_k^T\mathbf{R}_0\mathbf{V}_k$. Notice that the size of both $\mathbf{Y}$ and $\mathbf{C}$ are $k \times k$, so the dimension of Equation (11) is typically small and we can solve it by Global Conjugate method [5].

Another subtle issue is how to choose the initial vectors $\mathbf{g}$ and $\mathbf{f}$ for the Arnoldi Process. We use a procedure in [6] to choose $\mathbf{f}$ and $\mathbf{g}$ to guarantee that $\mathbf{r}_0 \in \mathcal{K}_k(\mathbf{A_1}, \mathbf{g}) \otimes \mathcal{K}_k(\mathbf{A_2}, \mathbf{f})$ as follows. If $||\mathbf{R}_0||_1 \le ||\mathbf{R}_0||_\infty$, we choose $\mathbf{f}$ as the column of $\mathbf{R}_0$ with the largest l_2 norm and $\mathbf{g} = \mathbf{R}_0^T\mathbf{f}/|\mathbf{f}|_2^2$; otherwise, we choose $\mathbf{g}$ as the row of $\mathbf{R}_0$ with the largest l_2 norm and $\mathbf{f} = \mathbf{R}_0\mathbf{g}/|\mathbf{g}|_2^2$.

Putting everything together, we have the overall algorithm for solving the Equation (1) in Algorithm 1.

Algorithm 1 FASTEN-P

Input: Normalized adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, tolerance parameter $\epsilon > 0$, Krylov subspace size $k > 0$, preference matrix $\mathbf{B}$;
Output: The solution $\mathbf{X}$ of Equation (1).

1: Initialize $\mathbf{X}$ and the residual matrix $\mathbf{R}$;
2: **while** $||\mathbf{R}||_F > \epsilon$ **do**
3: Choose Arnoldi vectors $\mathbf{g} \in \mathsf{R}^n$ and $\mathbf{f} \in \mathsf{R}^n$;
4: Apply Arnoldi Process on $\mathbf{A}_1$, $\mathbf{A}_2$ and obtain $\tilde{\mathbf{H}}_1$, $\tilde{\mathbf{H}}_2$, $\mathbf{V}_k$, $\mathbf{V}_{k+1}$, $\mathbf{W}_k$, $\mathbf{W}_{k+1}$;
5: Use Global Conjugate Gradient method for the linear system $\mathscr{L}(\mathbf{Y}) = \mathbf{C}$;
6: Update solution $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{V}_k\mathbf{Y}\mathbf{W}_k^T$;
7: Update residual $\mathbf{R} \leftarrow \mathbf{R} - \mathbf{V}_{k+1}\tilde{\mathbf{H}}_1\mathbf{Y}\tilde{\mathbf{H}}_2^T\mathbf{W}_{k+1}^T + \mathbf{V}_k\mathbf{Y}\mathbf{W}_k^T$;
8: **end while**

The first line is the initialization of the solution matrix and the residual. Line 2 to 8 are the outer loop, while line 5 is the inner loop which uses the Global Conjugate Gradient method. Line 6 (Equation (7)) updates the solution matrix $\mathbf{X}$ at each iteration. Line 7 (Equation (8) and Equation (9)) updates the residual matrix $\mathbf{R}$ at each iteration.

3.3 Proposed FASTEN-P+ Algorithm

To further reduce the time and space complexity of FASTEN-P, we explore the low-rank structure of the preference matrix $\mathbf{B}$ of Equation (1). Firstly, it is common in many graph mining tasks that the matrix $\mathbf{B}$ of Equation (1) has a low-rank structure. For example, in network alignment, anchor links are often sparse (e.g. in Figure 1, only 1 anchor link is given). If the prior knowledge of anchor links is unknown, matrix $\mathbf{B}$ becomes a uniform matrix, which means it is rank-1. Secondly, it turns out that the low-rank structure of the preference matrix $\mathbf{B}$ would imply the solution matrix $\mathbf{X}$ must have a low-rank block matrix structure (see the detailed analysis and proof in Lemma 3.2). This allows us to implicitly represent both the residual matrix and the solution matrix, which leads to a *linear* complexity.

The improved algorithm (FASTEN-P+) is summarized in Algorithm 2. As we can see the residual $\mathbf{R}$ is implicitly represented by

the multiplication of $\mathbf{U}_1$ and $\mathbf{U}_2$, and the solution $\mathbf{X}$ is implicitly represented by the multiplication of $\mathbf{M}$ and $\mathbf{N}$. The residual is updated at each iteration in line 9 and the solution is updated in line 7. Due to the implicit representation of the initial residual matrix, we need a slightly different procedure to choose the Arnoldi vector in line 4. Given the implicit representation of residual by $\mathbf{U}_1$ and $\mathbf{U}_2$, let $\mathbf{r}_1 = \mathbf{e}^{\mathbf{T}}\mathbf{U}_1\mathbf{U}_2$, $\mathbf{r}_2 = \mathbf{U}_1\mathbf{U}_2\mathbf{e}$ ($\mathbf{e}$ is an all-one vector), and let i_1 and i_2 be the indexes of the largest entries in $\mathbf{r}_1$ and $\mathbf{r}_2$. If $\max(\mathbf{r}_1) \geq \max(\mathbf{r}_2)$, we choose $\mathbf{f}$ as $\mathbf{U}_1\mathbf{U}_2(:, i_1)$, and then $\mathbf{g} = \mathbf{U}_2^T\mathbf{U}_1^T\mathbf{f}/|\mathbf{f}|_2^2$; otherwise, we set $\mathbf{g}$ as $\mathbf{U}_2^T\mathbf{U}_1(i_2, :)^T$, and then $\mathbf{f} = \mathbf{U}_1\mathbf{U}_2\mathbf{g}/|\mathbf{g}|_2^2$.

Algorithm 2 FASTEN-P+

Input: Normalized adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, tolerance parameter $\epsilon > 0$, Krylov subspace size $k > 0$, preference matrix $\mathbf{B}$;

Output: The implicit representation for the solution matrix $\mathbf{X}$ of Equation (1): $\mathbf{P}$ and $\mathbf{Q}$.

1: Initialize $\mathbf{P}, \mathbf{Q}$. Set $\mathbf{e}$ be an all-one vector.;
2: Let $\mathbf{U}_1 \leftarrow \mathbf{B}(:, 1)$, $\mathbf{U}_2 \leftarrow \mathbf{e}$;
3: **while** trace$(\mathbf{U}_2{}^T(\mathbf{U}_1{}^T\mathbf{U}_1)\mathbf{U}_2) > \epsilon$ **do**
4: Choose Arnoldi vectors $\mathbf{g} \in \mathsf{R}^n$ and $\mathbf{f} \in \mathsf{R}^n$;
5: Apply Arnoldi Process on $\mathbf{A}_1$ and $\mathbf{A}_2$ to obtain $\tilde{\mathbf{H}}_1$, $\tilde{\mathbf{H}}_2$, $\mathbf{V}_k$, $\mathbf{V}_{k+1}$, $\mathbf{W}_k$, $\mathbf{W}_{k+1}$;
6: Use Global Conjugate Gradient method for $\mathscr{L}(\mathbf{Y}) = \mathbf{C}$;
7: Construct $\mathbf{P} \leftarrow [\mathbf{P}, \mathbf{V}_k\mathbf{Y}], \mathbf{Q} \leftarrow [\mathbf{Q}, \mathbf{W}_k^T]$;
8: $\mathbf{L}_2 \leftarrow \mathbf{V}_{k+1}\tilde{\mathbf{H}}_1\mathbf{Y}\tilde{\mathbf{H}}_2^T$, $\mathbf{P}_2 \leftarrow \mathbf{W}_{k+1}^T$, $\mathbf{L}_3 \leftarrow \mathbf{V}_k\mathbf{Y}$, $\mathbf{P}_3 \leftarrow \mathbf{W}_k^T$;
9: Construct $\mathbf{U}_1 \leftarrow [\mathbf{U}_1, \mathbf{L}_2, \mathbf{L}_3]$, $\mathbf{U}_2 \leftarrow [\mathbf{U}_2{}^T, \mathbf{P}_2^T, \mathbf{P}_3^T]^T$;
10: **end while**
11: Return $\mathbf{P}, \mathbf{Q}$.

From line 3 to line 10 are the outer loop where the algorithm updates $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{U}_1$, $\mathbf{U}_2$ and checks stopping condition. Line 6 is the inner loop of Global Conjugate Gradient method.

We remark that the low-rank representation for the solution matrix $\mathbf{X}$ in Algorithm 2 bears subtle difference from [7, 20, 21], which *approximate* the input adjacency matrices with an inevitable approximation error. In contrast, the low-rank representation of the solution matrix in the proposed FASTEN-N is due to the low-rank structure of the preference matrix $\mathbf{B}$, regardless whether or not $\mathbf{A}_1$ and $\mathbf{A}_2$ are low-rank. As such, it does not introduce any approximation error.

3.4 Proofs and Analysis

3.4.1 Correctness. The correctness of FASTEN-P is summarized in Theorem 3.1, which says the matrix $\mathbf{X}$ by Algorithm 1 finds the exact solution of Equation (1).[2]

Theorem 3.1 (Correctness of FASTEN-P). *The matrix $\mathbf{X}$ by Algorithm 1 is the exact solution of Equation* (1) *w.r.t the tolerance ϵ.*

Proof. First, we prove that $\mathbf{V}_k \otimes \mathbf{W}_k$ is the orthonormal basis of $\mathcal{K}_k(\mathbf{A}_1, g) \otimes \mathcal{K}_k(\mathbf{A}_2, f)$. The Hessenberg matrix $\mathbf{H_1} \otimes \mathbf{H_2}$ and $\mathbf{A}_1 \otimes \mathbf{A}_2$ satisfy the following relationship

$$\begin{aligned}\mathbf{H_1} \otimes \mathbf{H_2} &= (\mathbf{V}_k^T\mathbf{A}_1\mathbf{V}_k) \otimes (\mathbf{W}_k^T\mathbf{A}_2\mathbf{W}_k) \\ &= (\mathbf{V}_k^T \otimes \mathbf{W}_k^T)(\mathbf{A}_1 \otimes \mathbf{A}_2)(\mathbf{V}_k \otimes \mathbf{W}_k) \\ &= (\mathbf{V}_k \otimes \mathbf{W}_k)^T(\mathbf{A}_1 \otimes \mathbf{A}_2)(\mathbf{V}_k \otimes \mathbf{W}_k)\end{aligned}$$

Therefore, $\mathbf{V}_k \otimes \mathbf{W}_k$ forms the orthonormal basis of $\mathcal{K}_k(\mathbf{A}_1, \mathbf{g}) \otimes \mathcal{K}_k(\mathbf{A}_2, \mathbf{f})$.

We then show that the least square problem with regard to $\mathbf{Y}$ in Equation (10) can be solved by a normal Equation (11).

We define the following linear operation: $\Phi(\mathbf{Y}) = \tilde{\mathbf{H}}_2\mathbf{Y}\tilde{\mathbf{H}}_1^T - \mathbf{I}_{k+1;k}\mathbf{Y}\mathbf{I}_{k+1;k}^T$. Then, we have that $||\mathbf{R}_1||_F = ||\mathbf{W}_{k+1}^T\mathbf{R}_0\mathbf{V}_{k+1} - \Phi(\mathbf{Y})||_F$ is minimized if and only if the following condition holds [6]:

$$\Phi^T(\mathbf{W}_{k+1}^T\mathbf{R}_0\mathbf{V}_{k+1} - \Phi(\mathbf{Y})) = 0$$

where $\Phi(\mathbf{Y})^T$ is the joint linear operation of $\Phi(\mathbf{Y})$. Therefore, we have that

$$\tilde{\mathbf{H}}_2^T(\mathbf{W}_{k+1}^T\mathbf{R}_0\mathbf{V}_{k+1} - \Phi(\mathbf{Y}))\tilde{\mathbf{H}}_1 - \mathbf{I}_{k+1;k}^T(\mathbf{W}_{k+1}^T\mathbf{R}_0\mathbf{V}_{k+1} - \Phi(\mathbf{Y}))\mathbf{I}_{k+1;k} = 0$$

which is equivalent to Equation (11).

Finally, we prove that Algorithm 1 gives the exact solution of Equation (2). For an arbitrary vector $\mathbf{v} \in \mathcal{K}_k(\mathbf{A}_1, \mathbf{g}) \otimes \mathcal{K}_k(\mathbf{A}_2, \mathbf{f})$, we have that $(\mathbf{I} - \mathbf{A}_1 \otimes \mathbf{A}_2)\mathbf{v}$ also belongs to $\mathcal{K}_k(\mathbf{A}_1, \mathbf{g}) \otimes \mathcal{K}_k(\mathbf{A}_2, \mathbf{f})$. This is because $(\mathbf{A}_1 \otimes \mathbf{A}_2)\mathbf{v} \in \mathcal{K}_k(\mathbf{A}_1, \mathbf{g}) \otimes \mathcal{K}_k(\mathbf{A}_2, \mathbf{f})$.

Therefore, the Kronecker subspace is invariant under matrix $\mathbf{I} - \mathbf{A}_1 \otimes \mathbf{A}_2$. According to [13], this is a sufficient condition for the solution obtained from any (oblique or orthogonal) projection method onto the Kronecker subspace to be exact. Thus, FASTEN-P gives the exact solution of Equation (1) or its equivalent linear system (Equation (2)), which completes the proof.

In order to prove the correctness of FASTEN-P+, we first give the following lemma, which says that the low-rank structure of the preference matrix $\mathbf{B}$ implies that the solution matrix $\mathbf{X}$ must be low-rank as well.

Lemma 3.2 (Low-rank structure of the Sylvester equation). *If the preference matrix $\mathbf{B}$ in Equation* (1) *is of low-rank r, then the rank of the solution matrix $\mathbf{X}$ of the Equation* (1) *is upper-bounded by pr, where p is the number of iterations in the outer loop in Algorithm 2.*

Proof. Omitted for brevity.

Based on Lemma 3.2 and following a similar process as the proof for Theorem 3.1, we can prove that Algorithm 2 also gives the exact solution of Equation (1).

Lemma 3.3 (Correctness of FASTEN-P+). *If $\mathbf{P}$ and $\mathbf{Q}$ are the two matrices returned by Algorithm 2, matrix $\mathbf{X} = \mathbf{PQ}$ is the exact solution of Equation* (1) *w.r.t the tolerance ϵ.*

Proof. Omitted for brevity.

3.4.2 Efficiency. The time and space complexity of FASTEN-P and FASTEN-P+ are summarized in Lemma 3.4. We can see that the proposed FASTEN-P has a quadratic complexity in both time and space, which is already better than state-of-the-art algorithms for solving Sylvester equations on plain graphs - they either produce an inexact solution or require a super-quadratic time complexity (See Table 1 for a comparison). Furthermore, the proposed FASTEN-P+ has a linear complexity in both time and space.

[2] Following the convention in scientific computing, we say a solution $\mathbf{X}$ is exact w.r.t. a tolerance parameter ϵ if the Frobenius norm of the difference between $\mathbf{X}$ and the solution by the direct method is less than ϵ. Throughout the paper, ϵ is set to be a very small number, e.g., $\epsilon = 10^{-7}$.

Lemma 3.4 (Complexity of FASTEN-P and FASTEN-P+). *The time and space complexity of FASTEN-P are $O((2k+4)pn^2)$ and $O(n^2)$ respectively. The time and space complexity of FASTEN-P+ are $O(kp(m+(r+2k+1)n))$ and $O(m+(r+2k+1)n)$ respectively, where p is the number of iterations in the outer loop and r is the rank of $\mathbf{B}$.*

Proof Omitted for brevity.

4 FAST ALGORITHMS FOR ATTRIBUTED GRAPHS

In this section, we present the attributed Sylvester equation solvers (FASTEN-N and FASTEN-N+) for Equation (3) and Equation (4). We start by introducing the intuition and key ideas, and then present the detailed algorithms, followed by some analysis in terms of the accuracy and complexity.

4.1 Intuition and Key Ideas

First, we cannot directly apply FASTEN-P or FASTEN-P+ to solve Equation (3) and (4). The main difficulty lies in the summation of Kronecker products on the left side of Equation (3), which should be first approximated by a single Kronecker product and itself could take $O(n^3)$ in time (nearest Kronecker product) [19]. To address this issue, the key observation is that the solution matrix $\mathbf{X}$ for the attributed Sylvester equation (i.e., Equation (4)) has a block-diagonal structure. This allows us to decompose the original Sylvester equations into *a set of* inter-correlated small-scaled Sylvester equations, which can be in turn solved by block coordinate descent methods (FASTEN-N). Second, by exploring the low-rank structure of the solution matrix (as we did in the FASTEN-P+ algorithm in Section 3), we will be able to obtain a linear algorithm to solve Equation (4). Let us explain this by the example in Figure 2. We can rewrite its attributed Sylvester equation as follows (Equation (12)). It can be seen that we only need to solve two inter-correlated Sylvester equations w.r.t two 2×2 diagonal blocks ($\mathbf{X}^{11}$ and $\mathbf{X}^{22}$), which can be in turn solved by an efficient *block coordinate descent* (*BCD*) [17] method. For the two off-diagonal blocks ($\mathbf{X}^{12}$ and $\mathbf{X}^{21}$), they are the same as the corresponding blocks in the preference matrix $\mathbf{B}$.

$$\mathbf{X}^{11} - [\mathbf{A}_2^{11}\mathbf{X}^{11}(\mathbf{A}_1^{11})^T + \mathbf{A}_2^{12}\mathbf{X}^{22}(\mathbf{A}_1^{12})^T] = \mathbf{B}^{11} \tag{12a}$$

$$\mathbf{X}^{22} - [\mathbf{A}_2^{21}\mathbf{X}^{11}(\mathbf{A}_1^{21})^T + \mathbf{A}_2^{22}\mathbf{X}^{22}(\mathbf{A}_1^{22})^T] = \mathbf{B}^{22} \tag{12b}$$

$$\mathbf{X}^{12} = \mathbf{B}^{12} \tag{12c}$$

$$\mathbf{X}^{21} = \mathbf{B}^{21} \tag{12d}$$

4.2 Proposed FASTEN-N Algorithm

In the general case, Equation (4) can be decomposed into a group of inter-correlated equations of block variables:

$$\mathbf{X}^{ii} - \sum_{q=1}^{l} \mathbf{A}_2^{iq}\mathbf{X}^{qq}(\mathbf{A}_1^{iq})^T = \mathbf{B}^{ii} \tag{13a}$$

$$\mathbf{X}^{ij} = \mathbf{B}^{ij} \tag{13b}$$

where $1 \le i,j \le l, i \neq j$. Since the off-diagonal blocks $\mathbf{X}^{ij}$ are the same as the corresponding blocks in $\mathbf{B}$, we will focus on solving Equation (13a) in the proposed algorithm, using a *BCD* method. The goal is to minimize the overall residual, namely the residual of Equation (4). In each iteration, one diagonal block variables will be updated with other blocks fixed. Let $\tilde{\mathbf{B}}_i = \mathbf{B}^{ii} + \sum_{j\neq i}^{l} \mathbf{A}_2^{ij}\mathbf{X}^{jj}(\mathbf{A}_1^{ij})^T$. We will solve the following equation in the i^{th} iteration in order to update $\mathbf{X}^{ii}$:

$$\mathbf{X}^{ii} - \mathbf{A}_2^{ii}\mathbf{X}^{ii}(\mathbf{A}_1^{ii})^T = \tilde{\mathbf{B}}_i \tag{14}$$

If we only treat $\mathbf{X}^{ii}$ as variables with all other diagonal blocks fixed, Equation (14) is a Sylvester equation *without* attributes, and thus can be efficiently solved by the proposed FASTEN-P algorithm in Section 3.

Algorithm 3 FASTEN-N

Input: Normalized adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, node attribute matrices $\mathbf{N}_1$ and $\mathbf{N}_2$, preference matrix $\mathbf{B}$, tolerance parameter $\epsilon > 0$, Krylov subspace size $k > 0$, number of node attribute l;
Output: The solution $\mathbf{X}$ of Equation (4).

1: Initialize each diagonal block variable $\mathbf{X}^{ii}, \forall 1 \le i \le l$, residual matrix $\mathbf{R}$;
2: Construct block matrices $\mathbf{A}_1^{ij}, \mathbf{A}_2^{ij}, \mathbf{B}^{ij}, \mathbf{B}^{ij}, \forall 1 \le i,j \le l$ by the node attribute matrices $\mathbf{N}_1, \mathbf{N}_2$;
3: **while** $||\mathbf{R}||_F > \epsilon$ **do**
4: **for** $i = 1, ..., l$ **do**
5: Compute $\tilde{\mathbf{B}}_i = \mathbf{B}^{ii} + \sum_{j\neq i}^{l} \mathbf{A}_2^{ij}\mathbf{X}^{jj}(\mathbf{A}_1^{ij})^T$;
6: Apply Algorithm 1 on Equation (14) to obtain $\mathbf{X}^{ii}$;
7: **end for**
8: Let $\mathbf{R} \leftarrow \sum_{j=1}^{l}(\mathbf{B}^{jj} - \mathbf{X}^{jj}) + \sum_{i=1}^{l}\sum_{j=1}^{l} \mathbf{A}_2^{ij}\mathbf{X}^{jj}(\mathbf{A}_1^{ij})^T$;
9: **end while**

The proposed FASTEN-N is summarized in the algorithm 3. Line 1 and 2 initialize the diagonal block variables, and the block matrices used in the decomposed set of equations. Line 3 to line 9 are the outer loop which uses the *BCD* method and checks the stopping condition. Line 6 is the inner loop of Algorithm 1. Line 8 updates the overall residual of Equation (4).

4.3 Proposed FASTEN-N+ Algorithm

In order to further reduce the time and space complexity of FASTEN-N, we explore a similar strategy as in FASTEN-P+ by the low-rank structure of the preference matrix $\mathbf{B}$. Here, we further represent each block matrix of $\mathbf{B}$ in its low-rank form, which allows to implicitly represent the solution block matrices $\mathbf{X}^{ii}$ in the low-rank forms when solving the Equation group (13a). The proposed algorithm FASTEN-N+ is summarized in Algorithm 4. Like FASTEN-N, it also uses the block coordinate descent method. A key difference between FASTEN-N and FASTEN-N+ lies in that, instead of calculating $\tilde{\mathbf{B}}_i$ directly, FASTEN-N+ represents it by two matrices $\tilde{\mathbf{B}}_i^1 = [\mathbf{B}^{ii}(:,1), \mathbf{A}_2^{ij}\mathbf{P}_i]$ and $\tilde{\mathbf{B}}_i^2 = [\mathbf{e}; \mathbf{Q}_i\mathbf{A}_1^{ij}], \forall 1 \le i \le l$, where $\mathbf{e}$ is an all-one vector, and $\mathbf{P}_i, \mathbf{Q}_i$ are the implicit representation of solution $\mathbf{X}^{ii}$.

In Algorithm 4, line 4 to line 10 are the outer loop which uses the *BCD* method and checks the stopping condition, and line 7 is the inner loop of Algorithm 2. In line 6, the revised preference matrix $\tilde{\mathbf{B}}_i$ in the equation group (13a) is represented by two low-rank matrices. In line 9, the residual of each equation in the equation group is indirectly represented as well.

Algorithm 4 FASTEN-N+

Input: Normalized adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, node attribute matrices $\mathbf{N}_1$ and $\mathbf{N}_2$, preference matrix $\mathbf{B}$, tolerance parameter $\epsilon > 0$, subspace size $k > 0$;

Output: The implicit solution $\mathbf{P}_i, \mathbf{Q}_i, \forall 1 \le i \le l$ of Equation 14.

Initialize $\mathbf{P}_i, \mathbf{Q}_i, \forall 1 \le i \le l$. Set $\mathbf{e}$ be an all-one vector;
Construct block matrices $\mathbf{A}_1^{ij}, \mathbf{A}_2^{ij}, \mathbf{B}^{ij}, \mathbf{B}^{ij}, \forall 1 \le i, j \le l$ by the node attribute matrices $\mathbf{N}_1$ and $\mathbf{N}_2$;
Let $\mathbf{U}_1^i \leftarrow \mathbf{B}^{ii}(:, 1)$, $\mathbf{U}_2^i \leftarrow \mathbf{e}$;
while $\prod_i^l \text{trace}((\mathbf{U}_2{}^i)^T((\mathbf{U}_1{}^i)^T\mathbf{U}_1{}^i)\mathbf{U}_2{}^i) > \epsilon$ **do**
  for $i = 1, ..., l$ **do**
    $\tilde{\mathbf{B}}_i^1 \leftarrow [\mathbf{U}_1^i, \mathbf{A}_2^{ij}\mathbf{P}_i]$, $\tilde{\mathbf{B}}_i^2 \leftarrow [\mathbf{U}_2^i; \mathbf{Q}_i\mathbf{A}_1^{ij}]$, $\forall 1 \le i \le l$;
    Apply Algorithm 2 on Equation (14) to obtain $\mathbf{P}_i$, $\mathbf{Q}_i$;
  end for
  $\mathbf{U}_1^i \leftarrow [\mathbf{U}_1^i, -\mathbf{P}_i, \mathbf{A}_2^{ij}\mathbf{P}_j]$, $\mathbf{U}_2^i \leftarrow [\mathbf{U}_2^i; -\mathbf{Q}_i; \mathbf{Q}_j(\mathbf{A}_1^{ij})^T]$, $\forall 1 \le i, j \le l$;
end while
Return $\mathbf{P}_i$ and $\mathbf{Q}_i$, $(i = 1, ..., l)$.

4.4 Proofs and Analysis

4.4.1 Correctness. The correctness of the proposed FASTEN-N and FASTEN-N+ is summarized in Theorem 4.1, which says that both algorithms find the exact solution of Equation (4).

Theorem 4.1 (Correctness of FASTEN-N and FASTEN-N+). *The solution matrix* $\mathbf{X}$ *by Algorithm 3 is the exact solution of Equation* (4) *w.r.t the tolerance* ϵ. *If* $\mathbf{P}_i$ *and* $\mathbf{Q}_i$, $(i = 1, ..., l)$ *are the matrices returned by Algorithm 4, matrix* $\mathbf{X} = diag(\mathbf{P}_1\mathbf{Q}_1, ..., \mathbf{P}_l\mathbf{Q}_l) + \prod_{i,j}^{l} \mathbf{B}^{(ij)}$ *is the exact solution of Equation* (4) *w.r.t the tolerance* ϵ.

Proof. Omitted for brevity.

4.4.2 Efficiency. The time and space complexity of FASTEN-N and FASTEN-N+ are summarized in Lemma 4.2. We can see that the proposed FASTEN-N has a quadratic complexity in both time and space. Furthermore, the proposed FASTEN-N+ has a linear complexity in both time and space (See Table 1 for comparison).

Lemma 4.2 (Complexity of FASTEN-N and FASTEN-N+). *The time and space complexity of FASTEN-N are* $O(p_2mn/l+(2k+4)p_2n^2/l)$ *and* $O(n^2 + m/l)$ *respectively. The time and space complexity of FASTEN-N+ are* $O((p_1km + p_1k(r + lk + k + 1)n)p_2l)$ *and* $O(m + (r + kp_1(l-1))n)$, *respectively.* p_1 *is the number of iterations of FASTEN-N or FASTEN-N+in the inner loop, and* p_2 *is the number of iterations of the outer loop, and* r *is the rank of the preference matrix* $\mathbf{B}$.

Proof. Omitted for brevity.

5 EXPERIMENTAL RESULTS

In this section, we present the experimental results. The experiments are designed to evaluate the efficiency, the effectiveness and the parameter sensitivity of the proposed family of algorithms (FASTEN).

5.1 Experimental Setup

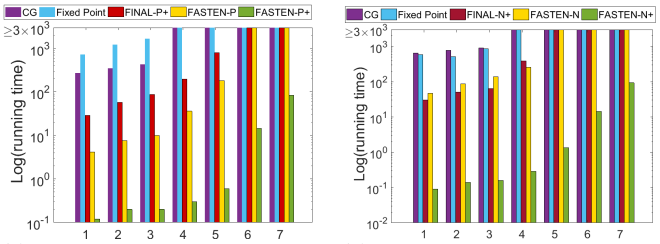
Datasets. We evaluate the proposed algorithms on five real-world datasets [16], which are summarized in Table 3.

Table 3: Datasets Summary

Dataset Name	Category	# of Nodes	# of Edges
DBLP	Co-authorship	9,143	16,338
Flickr	User relationship	12,974	16,149
LastFm	User relationship	15,436	32,638
AMiner	Academic network	1,274,360	4,756,194
LinkedIn	Social network	6,726,290	19,360,690

- *DBLP:* This is a co-authorship network with each node representing an author. Each author is assigned one node attribute vector of the number of publications in 29 major conferences [12].
- *Flickr:* This is a network of friends on the image and video hosting website *Flickr*. Node attribute vector is constructed from users' profile information (e.g. age, gender, location, etc.) [23].
- *LastFm:* Collected in 2013, this is the following network of users on the music website *LastFm* [23]. A detailed profile of users is provided. The node attribute vector is also constructed from users' profile information.
- *AMiner:* AMiner dataset represents the academic social network. Undirected edges represent co-authorship and the node attribute vector is extracted from the number of published papers [23].
- *LinkedIn:* The graph of *LinkedIn* dataset is from users' connections in the social network *LinkedIn*. The node attribute vector is constructed from users' profile information (e.g. age, gender, occupation, etc.)

Comparison Methods. We compare the proposed methods against four baseline methods, the *Fixed Point* method (*FP*) [18], the *Conjugate Gradient Method* (*CG*) [18], *FINAL-P+* [20], and *FINAL-N+* [20]. According to [19], *CG* is best known method in terms of efficiency to obtain the exact solution of the Sylvester equation studied in this paper. *FP* is widely used in solving Sylvester equation and linear system. *FINAL-P+* and *FINAL-N+* are two recent approximate methods that solve the Sylvester equation on plain graph and attributed graph respectively [20]. We set the rank of the input networks to 2 in both *FINAL-P+* and *FINAL-N+* to ensure a fast computation, and the runtime of these two methods with a higher rank would be



(a) Log of running time on different datasets. Lower is better. **(b) Log of running time on different datasets. Lower is better.**

Figure 3: Efficiency comparison on plain and attributed networks. Best viewed in color. Datasets: 1: *DBLP*, 2: *Flickr*, 3: *LastFm*, 4: *Aminer* with $25K$ nodes, 5: *Aminer* with $100K$ nodes, 6: *Aminer*, and 7: *LinkedIn*.

longer than the results reported below. We terminate an algorithm if it does not finish within 3,000 seconds.

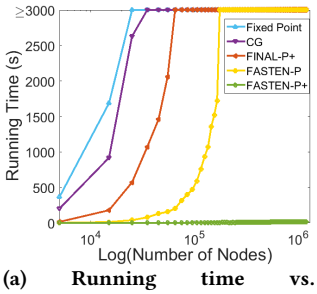
Repeatability. All datasets are publicly available. We will release the code of our proposed algorithms upon the publication of the paper. All experiments are performed on a server with 64 Intel(R) Xeon(R) CPU cores at 2.00 GHz and 1.51 TB RAM. The operating system is Red Hat Enterprise Linux Server release 6.9. All codes are written in MATLAB R2017a using a single thread.

5.2 Efficiency

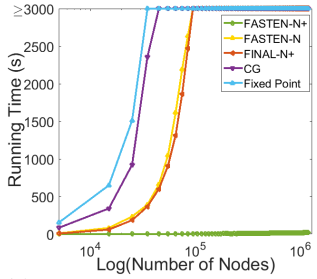
A - Speedup. We first evaluate how much speedup the proposed algorithms can achieve over baseline methods, on seven real-world datasets, including two subsets of *AMiner* with 25K and 100K nodes respectively. The results are presented in Figure 3.

As we can see from Figure 3(a), the proposed FASTEN-P already outperforms all the baselines on all datasets even including the approximate method *FINAL-P+*. The proposed FASTEN-P+ outperforms all baseline methods as well as FASTEN-P by a large margin. When it is applied on the largest dataset *LinkedIn*, the running time of FASTEN-P+ is less than 100 seconds; whereas all the other methods cannot finish within 3,000 seconds. On *AMiner* (with 25K nodes) dataset, FASTEN-P+ is more than 10,000× faster than *CG* (3,000+ seconds vs. 0.3 seconds).

On the attributed networks (Figure 3(b)), FASTEN-N outperforms *CG* and *FP*, and its running time is close to *FINAL-N+* although the latter produces an approximate solution. The proposed FASTEN-P+ outperforms all baseline methods as well as FASTEN-N by a large margin. On *AMiner* (with 25K nodes) dataset, FASTEN-N+ is more than 10,700× faster than *CG*. (3,000+ seconds vs. 0.28 seconds) .



(a) Running time vs. Log(number of nodes). Dataset: Aminer without attributes



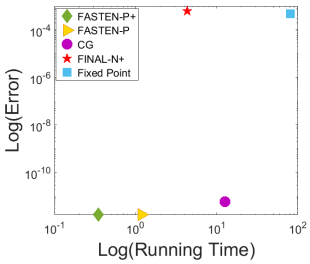
(b) Running time vs. Log(number of nodes). Dataset: Aminer with attributes

Figure 4: Scalability on plain (left) and attributed graphs (right). Best viewed in color.

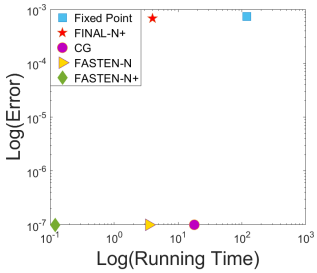
B - Scalability. The scalability experiments are conducted on the largest *Aminer* dataset. We run experiments on graphs with different size (ranging from 5K to 1.2M nodes) 100 times, and report the average running time. The results are presented in Figure 4. We can see that all the baseline methods are at least quadratic w.r.t the number of nodes in graphs, and could not finish within 3,000 seconds for graphs with more than 100,000 nodes. Both the proposed FASTEN-P+ and FASTEN-N+ scale linearly to million-node graphs.

5.3 Effectiveness

For the effectiveness evaluation, we define the error of the solution as: $Error = ||\mathbf{X} - \mathbf{X}'||_F$ where $\mathbf{X}$ is the solution matrix by FASTEN or other baseline methods, and $\mathbf{X}'$ is computed by the direct method on the equivalent linear system of the Sylvester equation (namely Equation (2) and (3)). Since the direct method takes $O(n^6)$ in time, we use a subset of *AMiner* with 4K nodes in this experiment to avoid extremely long running time of the direct method. We compare the *Error* vs. the running time of our methods with all the baseline methods in Figure 5. We can observe that the *Conjugate Gradient (CG)* method and all of our proposed FASTEN algorithms have a very small *Error* (less than 10^{-7}). *Error* of both *Fixed Point* and *FINAL-N+* are more than 10^{-4}. In the meanwhile, the running time of the proposed FASTEN is smaller than all baseline methods in all cases.



(a) log of error vs. log of running time ($\epsilon = 10^{-12}$; $k = 14$)



(b) log of error vs. log of running time ($\epsilon = 10^{-7}$; $k = 5$)

Figure 5: Error vs. running time comparison of five methods on plain graphs (left) and attributed graphs (right). Best viewed in color.

5.4 Parameter Sensitivity

In the proposed FASTEN algorithms, we need to set Krylov subspace size k, which might affect the convergence rate and the running time of the proposed algorithms. Generally speaking, a smaller k makes the computation of the inner loop faster, but might cause a slower convergence of the outer loop (see Section 3 and Section 4 for the detailed descriptions of inner loop and outer loop); on the other hand, it would take longer time for the inner-loop with a larger k although it might help reduce the iteration number of the outer-loop . Take FASTEN-P as an example, we report the running time of FASTEN-P vs. the subspace size k on three datasets in Figure 6. We can see the running time stays stable at a low number when $14 \leq k \leq 60$, and it starts to increase when k is outside this range. Overall, we found that the running time of all the four proposed algorithms is insensitive in a relatively large range of the Krylov subspace size k.

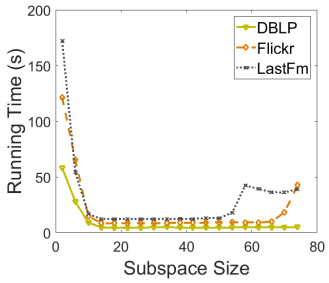


Figure 6: Sensitivity study of FASTEN-P. Best viewed in color.

6 RELATED WORK

A variety of graph mining tasks (e.g. interactive subgraph matching [4], local graph cut [24], etc.) have been studied recently by

novel methods. The Sylvester equation is commonly used as a core building block. Vishwanathan et al. in [18] compare four algorithms for Sylvester equation in the context of random walk graph kernel, including the Fixed Point iterative method (*FP*), the Conjugate Gradient method (*CG*), one Sylvester equation solver (*Sylv.*) and the direct method. *ARK* by Kang et al. in [7] applies the top-r eigen-decomposition on the adjacency matrices to approximate graph kernel with lower complexity. *Cheetah* [10] by Li et al. tracks the graph kernel by incrementally updating the low-rank approximation on input graphs. For the task of network alignment, *FINAL* by Zhang et al. [20] propose a fixed-point as well as an approximate method to solve the attributed Sylvester equation. For subgraph matching, *FIRST* by Du et al. [4] propose an approximate method for solving the Sylvester equation in the scenario of interactive subgraph matching.

The Sylvester equation has been studied for a long time in the applied mathematics and scientific computing community, and has applications in many domains. The classic full orthogonal method *FOM* [15], conjugate gradient method *CG* [14] and global minimal residual method *GMRES* [13] are summarized by Saad et al. in [13]. Recently, many studies on the generalized Sylvester equation are proposed. Ke et al. propose a preconditioned nested splitting conjugate gradient iterative method [8] for large sparse generalized Sylvester equation. Bao et al. propose a Galerkin and minimal residual method [1] for iteratively solving a generalized Sylvester equation which has a similar formation as our problem. Beik et al. propose global Krylov subspace methods [2] for solving general coupled matrix equations, which can be seen as a generalized version of the Sylvester equation in our setting. In [3], Bouhamidi et al. propose a global-GMRES which uses modified global Arnoldi algorithm to solve the linear matrix equation of the generalized Sylvester equation.

7 CONCLUSION

In this paper, we propose a family of Krylov subspace based algorithms (FASTEN) to speed up and scale up the computation of Sylvester equation for graph mining. The key idea is to project the original equivalent linear system onto a Kronecker Krylov subspace. Based on that, we propose the following methods to further reduce complexity, including (1) implicitly representing the solution based on its low-rank structure, and (2) decomposing the original Sylvester equation into a set of small-scale, inter-correlated Sylvester equations. The proposed algorithms bear two distinctive features. First, they provide the *exact* solutions without any approximation error. Second, they significantly reduce the time and space complexity for solving Sylvester equation. Two of the proposed algorithms have *linear* time and space complexity. We conduct numerous experiments on real-world data, and show that the FASTEN family of algorithms (i) produce the exact solution, (ii) are up to more than $10,000\times$ faster than the best known method, and (iii) scale up to million-node graphs in about 100 seconds.

8 ACKNOWLEDGEMENTS

This material is supported by the National Science Foundation under Grant No. IIS-1651203, IIS-1715385, IIS-1743040, and CNS-1629888, by DTRA under the grant number HDTRA1-16-0017, by the United States Air Force and DARPA under contract number FA8750-17-C-0153 , by the U.S. Department of Homeland Security under Grant Award Number 2017-ST-061-QA0001 and by Army Research Office under the contract number W911NF-16-1-0168. The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] Liang Bao, Yiqin Lin, and Yimin Wei. 2006. Krylov subspace methods for the generalized Sylvester equation. *Applied mathematics and computation* (2006).
[2] Fatemeh Panjeh Ali Beik and Davod Khojasteh Salkuyeh. 2011. On the global Krylov subspace methods for solving general coupled matrix equations. *Computers & Mathematics with Applications* (2011).
[3] Abderrahman Bouhamidi and Khalide Jbilou. 2008. A note on the numerical approximate solutions for generalized Sylvester matrix equations with applications. *Appl. Math. Comput.* (2008).
[4] Boxin Du, Si Zhang, Nan Cao, and Hanghang Tong. 2017. First: Fast interactive attributed subgraph matching. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.
[5] A El Guennouni, Khalide Jbilou, and AJ Riquet. 2002. Block Krylov subspace methods for solving large Sylvester equations. *Numerical Algorithms* (2002).
[6] Dan Y Hu and Lothar Reichel. 1992. Krylov-subspace methods for the Sylvester equation. *Linear Algebra Appl.* (1992).
[7] U Kang, Hanghang Tong, and Jimeng Sun. 2012. Fast random walk graph kernel. In *Proceedings of the 2012 SIAM International Conference on Data Mining*. SIAM.
[8] Yi-Fen Ke and Chang-Feng Ma. 2014. A preconditioned nested splitting conjugate gradient iterative method for the large sparse generalized Sylvester equation. *Computers & Mathematics with Applications* (2014).
[9] Cuiping Li, Jiawei Han, Guoming He, Xin Jin, Yizhou Sun, Yintao Yu, and Tianyi Wu. 2010. Fast computation of simrank for static and dynamic information networks. In *Proceedings of the 13th International Conference on Extending Database Technology*.
[10] Liangyue Li, Hanghang Tong, Yanghua Xiao, and Wei Fan. 2015. Cheetah: fast graph kernel tracking on dynamic graphs. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM.
[11] Kensuke Onuma, Hanghang Tong, and Christos Faloutsos. 2009. TANGENT: a novel,'Surprise me', recommendation algorithm. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*.
[12] Adriana Prado, Marc Plantevit, Céline Robardet, and Jean-Francois Boulicaut. 2013. Mining graph topological patterns: Finding covariations among vertex descriptors. *Knowledge and Data Engineering, IEEE Transactions on* (2013).
[13] Yousef Saad. 2003. *Iterative methods for sparse linear systems*. SIAM.
[14] Jonathan Richard Shewchuk et al. 1994. An introduction to the conjugate gradient method without the agonizing pain. (1994).
[15] Kunio Tanabe. 1971. Projection method for solving a singular system of linear equations and its applications. *Numer. Math.* (1971).
[16] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*.
[17] Paul Tseng. 2001. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications* (2001).
[18] SVN Vishwanathan, Karsten M Borgwardt, and Nicol N Schraudolph. 2006. Fast computation of graph kernels. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*.
[19] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. *Journal of Machine Learning Research* (2010).
[20] Si Zhang and Hanghang Tong. 2016. FINAL: Fast Attributed Network Alignment.. In *KDD*. 1345–1354.
[21] Si Zhang, Hanghang Tong, Jie Tang, Jiejun Xu, and Wei Fan. 2017. iNEAT: Incomplete Network Alignment. In *2017 IEEE International Conference on Data Mining (ICDM)*.
[22] Yongfeng Zhang. 2014. Browser-oriented universal cross-site recommendation and explanation based on user browsing logs. In *Proceedings of the 8th ACM Conference on Recommender systems*.
[23] Yutao Zhang, Jie Tang, Zhilin Yang, Jian Pei, and Philip S Yu. 2015. Cosnet: Connecting heterogeneous social networks with local and global consistency. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
[24] Dawei Zhou, Si Zhang, Mehmet Yigit Yildirim, Scott Alcorn, Hanghang Tong, Hasan Davulcu, and Jingrui He. 2017. A local algorithm for structure-preserving graph cut. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.