

https://dspace.org/

Maximum edge biclique via matrix decomposition

Salma Omer; Montaz Ali

AMER INST MATHEMATICAL SCIENCES
https://hdl.handle.net/10539/47318
WOS

Downloaded from a DSpace repository. https://dspace.org/

Journal of Industrial and Management Optimization
Vol. 21, No. 11, November 2025, pp. 6270-6294
doi:10.3934/jimo.2025131

MAXIMUM EDGE BI-CLIQUE VIA MATRIX DECOMPOSITION

Salma Omer[✉*1] and Montaz Ali[✉1]

[1]School of Computer Science and Applied Mathematics,
University of Witwatersrand, South Africa

(Communicated by Gerhard-Wilhelm Weber)

Abstract. The maximum edge bi-clique (MEB) problem is the problem of finding a bi-clique with the maximum number of edges in a bipartite graph. In this paper, we have proposed a mathematical model for MEB along with an ADMM (Alternating Direction Method of Multiplier) based algorithm. Although the constraint set of the mathematical model resembles the matrix decomposition involving the adjacency matrix of a given graph, we have included the weighted ℓ_1-norm in the objective function. We provide an analysis showing how the weighted ℓ_1-norm induces rapid sparsity and, therefore, faster convergence of the ADMM algorithm. In addition, we have suggested a 'natural' condition with which to identify the MEB in random graphs. We have tested our mathematical model by running the ADMM algorithm on planted bi-cliques as well as on randomly generated graphs. Numerical comparisons with a recent algorithm using planted bi-cliques have shown the superiority of our approach. The errors obtained for random graphs have shown near-exact recoveries. Finally, we have applied our algorithm to real-world bipartite graphs for which bi-cliques have been recovered successfully.

1. **Introduction.** Bi-clique is a fundamental structure of bipartite graphs. This structure is widely used in a variety of bipartite applications to capture cohesive sub-graphs. Real-world applications of bi-cliques span numerous fields. In social network analysis, bi-cliques can represent tightly connected communities or interest groups between users and content. In bioinformatics, they can model interactions between genes and conditions in gene expression data. In e-commerce platforms, bi-cliques can capture frequent co-purchasing patterns between customers and products. Similarly, in recommendation systems, they can highlight subsets of users and items with strong mutual affinities. In cybersecurity, detecting suspicious user-resource interactions, such as dense bipartite sub-graphs, is key to anomaly detection.

Consequently, solving the Maximum Edge Bi-Clique (MEB) problem has broad practical implications for pattern discovery, prediction, and anomaly detection in real-world systems with inherently bipartite structures.

2020 *Mathematics Subject Classification.* Primary: 65Kxx, 90Cxx; Secondary: 90C25, 90C27, 90C35.

Key words and phrases. Maximum edge bi-clique, convex relaxation, matrix decomposition, integer solution, ADMM.

*Corresponding author: Salma Omer.

© 2025 The Author(s). Published by AIMS, LLC. This is an Open Access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

A bipartite graph (or bi-graph) consists of vertices that can be separated into two independent and disjoint sets. A bi-clique in such a graph is a complete bipartite sub-graph.

Let $G_b = (V, E) = (V_1 \cup V_2, E \subseteq (V_1 \times V_2))$, where V represents the set of vertices and E represents the set of edges of G_b, be a bipartite graph with two independent and disjoint sets, V_1 and V_2, such that every edge links a vertex in V_1 to one in V_2.

Note that a bi-clique (V_1^*, V_2^*) has $|V_1^*| + |V_2^*|$ vertices and $|V_1^*| \times |V_2^*|$ edges. When $|V_1^*| = |V_2^*|$, then the problem is also called balanced bi-clique problem.

A bi-clique is defined as maximum if it is of the largest size, measured either by the number of vertices (vertex-maximum), known as the maximum vertex bi-clique problem, or by the number of edges (edge-maximum), known as the MEB problem. In the maximum balanced bi-clique problem the goal is to find a complete bipartite graph with a maximum number of vertices, and it has been proved to be NP-complete; see [15]. Peeters [21] has shown that the MEB problem is NP-complete. On the other hand, [15] has shown that the maximum vertex bi-clique problem is solvable in a polynomial time when the input graph is bipartite. However, the problem is NP-complete when the input graph is not bipartite; see [24].

In this paper, we propose a novel mathematical formulation for the MEB problem based on matrix decomposition, which integrates a weighted ℓ_1-norm in the objective function. Unlike traditional formulations that treat low-rank and sparse components with uniform penalties, our approach adaptively penalizes the sparse component using a dynamically updated weight matrix, enabling faster sparsity induction and improved convergence. This structure not only enhances the quality of bi-clique recovery, but also promotes near-binary solutions, making the output more interpretable and practically useful. The proposed model is solved using a customized Alternating Direction Method of Multipliers (ADMM) algorithm designed to exploit the separability of the objective while incorporating adaptive weighting. Theoretical analysis establishes conditions under which exact recovery is guaranteed, while extensive experiments on both synthetic and real-world bipartite graphs demonstrate that our approach achieves superior performance in terms of accuracy, recovery rate, and computational efficiency when compared to recent baseline methods.

The remaining sections of the paper are organized as follows. Section 2 presents the related approaches to the MEB problem. Section 3 presents the model for MEB and the ADMM algorithm for the proposed model. Then we present the theoretical recovery and the uniqueness of the solution in Section 4. In Section 5, we evaluate the performance of the proposed algorithm for recovering the planted bi-clique, recovering the maximum bi-clique from randomly generated graphs, and also the maximum bi-cliques from real-world bipartite graphs. Finally, we make our final remarks in Section 6.

2. **Related work.**

Lyu et al. [17] have proposed a progressive bounding framework to find the bi-clique with the maximum number of edges in large-scale real-world datasets. They have also extended the branch-and-bound algorithm for the MEB problem.

A rank-one non-negative factorization model for the MEB has been proposed by [12]. In this formulation a bi-clique is constructed by minimizing the number of edges outside the bi-clique instead of maximizing the number of edges inside. Gillis and Glineur [13] have also proposed an approximate rank-one matrix factorization problem as a continuous characterization of the MEB problem.

The MEB problem is known to have no good approximation algorithm. However, based on some assumptions, Ambühl et al. [2] have obtained hardness results for the MEB problem. The paper [18] presents conditional inapproximability results for the MEB problem based on the expansion hypothesis of a small set.

Currently, the majority of algorithms that solve the MEB problem enumerate all maximal bi-cliques in the given graph; see [16, 1] and [11]. Nussbaum et al. [19] have developed a pruning strategy that avoids enumerating all maximal bi-cliques.

Recently, [23] have introduced an algorithm based on integer programming to determine the MEB in bipartite graphs. In a recent paper, [6] have proposed the densest sub-graph algorithm to identify the densest subgraph and densest sub-matrix problems, which can be seen as a generalization of the MEB problem. However, the main drawback of these algorithms is that the MEB size must be known a priori.

A nuclear norm heuristic has also been proposed for the planted bi-clique problem; see [3] and [4]. It has been shown that when the given bipartite graph contains a planted bi-clique of sufficiently large size, the proposed heuristic is exact.

Recent developments in bi-clique enumeration and matrix decomposition have advanced the state-of-the-art in dense sub-graph detection and graph learning. Chen et al. [10] proposed an efficient algorithm for bi-clique counting in large bipartite graphs, achieving scalability through combinatorial optimizations. Zhou et al. [26] presented a convex programming-based framework for anchored densest subgraph search, which is conceptually related to the maximum edge bi-clique (MEB) problem. From a decomposition perspective, Bertsimas et al. [5] developed a discrete optimization method for sparse plus low-rank matrix decomposition, which provides theoretical guarantees for exact recovery under certain conditions. Furthermore, Zhai and Zhang [25] introduced a learnable graph-regularized matrix decomposition framework that adaptively encodes structural information, showing promise for graph-based data recovery. Our work builds on these insights by formulating the MEB problem as a convex optimization task involving low-rank and sparse components, guided by a dynamically weighted ℓ_1-norm.

3. Our approach to maximum bi-clique.

We propose a matrix decomposition approach to mathematically model the MEB problem. The matrix decomposition has a number of applications in image analysis and computer vision, see [22].

We have taken the matrix decomposition approach for the planted bi-clique problem. The matrix decomposition problem separates a given matrix M_b into a low-rank and a sparse component by solving the problem.

$$\begin{aligned} &\min_{L,S\in\mathbb{R}^{N\times M}} \ \operatorname{rank}(L)+\lambda\|S\|_0 \\ &s.t.\ L+S=M_b, \end{aligned}$$

where $\|S\|_0 = \operatorname{card}(S)$ is the number of non-zero entries in S; λ is the regularization parameter;. Both the rank function and ℓ_0-norm minimization are non-convex. The nuclear norm $\|L\|_*$ is the sum of singular values $\sigma_i(L)$; it is used as the convex relaxation of the rank function, and the ℓ_1-norm, $\|S\|_1 = \sum_{i=1}^{N}\sum_{i=1}^{M}|S_{ij}|$, is used as the convex relaxation of $\|S\|_0$.

Given a bipartite graph $G_b = (V_1 \cup V_2, E)$, $|V_1| = N$ and $|V_2| = M$, the adjacency matrix M_b of G_b is an $N \times M$ matrix of zeros and ones that represents the relations between the vertices of G_b. In the context of the bi-clique problem, if we include self-loops and assign 1 to the diagonal elements of the adjacency matrix M_b, then M_b can be split into a rank-one matrix L (corresponding to the maximum bi-clique)

and a sparse matrix S. Here, we assume that the bi-clique is of size $n \times m$. Hence; matrix L of rank one corresponds to the bi-clique with block of size $n \times m$ with entry values one and remaining entries zeros. The entries of M_b that correspond to the edges outside the bi-clique go to S. Thus, we consider the following convex formulation for MEB and the planted bi-clique problems:

$$\min_{L,S \in \mathbb{R}^{N \times M}} \ \|L\|_* + \lambda \|S\|_1 \tag{1}$$

$$s.t.\ M_b - L - S = 0, \tag{2}$$

$$S_{ij} \in [0, 1], \tag{3}$$

where M_b is the input adjacency matrix; L and S are the optimization variables; $\|L\|_*$ is the nuclear norm defined as the sum of singular values $\sigma_i(L)$, the ℓ_1-norm $\|S\|_1$ is defined as $\sum_{j=1}^{M} \sum_{i=1}^{N} |S_{ij}|$. We refer to problem (1)-(3) as the 'regular' matrix decomposition formulation for the planted bi-clique problem.

The main difficulty of the above model is that the entries of optimal L and S may not be integers. In fact, this was observed for the nuclear norm minimization model by Ames [3] who rounded each entry of the optimal L to the nearest integers. The rounding of entries causes noisy recovery, as indicated by a number of figures presented in [3]. In the following, we propose our mathematical model that can overcome the above difficulty. The main concept of our approach is to use the weighted $\|S\|_1$ norm in the objective function (1). We have demonstrated a posteriori that the weighted $\|S\|_1$ norm is central to achieving the integer value of the entries of L and S. We have taken a systematic approach to generate the weights. We approximating each term of $\|S\|_0$,

$$\|S\|_0 = \# \begin{cases} 1 & \text{if } S_{ij} \neq 0, \\ 0 & \text{otherwise,} \end{cases}$$

with the function,

$$\psi(S_{ij}) = \frac{S_{ij}}{S_{ij} + \rho}, S_{ij} \neq -\rho,\ \rho > 0. \tag{4}$$

The function ψ is concave in $[0, \infty)$ with $\psi'(S_{ij}) > 0$ for all $S_{ij} \geq 0$, $\psi'(0) < \infty$.

Before presenting the mathematical model, we make a graphical comparison of $\psi(x)$ in (4) with other approximations of $\|x\|_0$ in the case of a single variable. The comparison in Figure 1 shows that $\psi(x)$, $\rho \to 0$, gives a better approximation of $\|x\|_0$ than $|x|$ and $\log(|x| + \rho)$. This motivates our choice of small values of ρ.

Our method extends recent work on sparse and low-rank decomposition, particularly the discrete optimization framework by Bertsimas et al. [5]. We adopt a dynamically weighted ℓ_1-norm to promote sparsity, drawing inspiration from adaptive graph-regularized models [25], while tailoring the formulation to the bi-clique recovery problem.

Hence, our choice of function $\psi(S_{ij})$ further validates the claim made in [9] that the ℓ_1-norm is not a good approximation for the ℓ_0-norm. Furthermore, in our mathematical model $S_{ij} = -\rho$ can be easily avoided, since $S_{ij} \in [0, 1]$. We write our relaxed objective function as $\|L\|_* + \lambda \Phi(S)$,

$$\Phi(S) = \sum_{i=1}^{N} \sum_{j=1}^{N} \psi(S_{ij}). \tag{5}$$

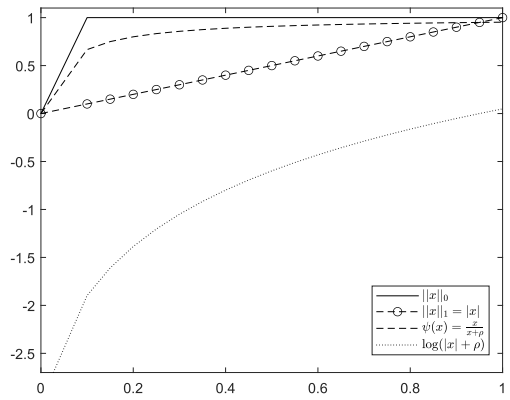


FIGURE 1. Comparison between $\psi(x)$, the penalty log function $\log(|x| + \rho)$, ℓ_1-norm and ℓ_0-norm in the case of single variable, $\rho = 0.05$.

We now construct a convex surrogate of $\Phi(S)$ in a known feasible S_{J-1}, say at $(J-1)$-th iteration of an algorithm. It follows from the concavity of ψ that

$$\psi(S_{ij}) \leq \psi((S_{J-1})_{ij}) + \psi'((S_{J-1})_{ij})(S_{ij} - (S_{J-1})_{ij}).$$

Hence, we have

$$\begin{aligned}\Phi(S) &= \sum_{i=1}^{N}\sum_{j=1}^{N}\psi(S_{ij}) \\ &\leq \sum_{i=1}^{N}\sum_{j=1}^{N}(\psi((S_{J-1})_{ij}) + \psi'((S_{J-1})_{ij})(S_{ij} - (S_{J-1})_{ij})) \\ &= \widetilde{\Phi}(S),\end{aligned}$$

where $\psi'((S_{J-1})_{ij}) = \frac{\epsilon}{((S_{J-1})_{ij}+\rho)^2}$. $\Phi(S)$ satisfies

$$\Phi(S) \leq \widetilde{\Phi}(S) \text{ and } \Phi(S_{J-1}) = \widetilde{\Phi}(S_{J-1}). \tag{6}$$

We now ignore the constant terms in $\widetilde{\Phi}(S)$ and treat the remaining expression as the surrogate for $\Phi(S)$. The concept of a surrogate function has been reported in [14]. Hence, the surrogate becomes

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\psi'((S_{J-1})_{ij})S_{ij} = \sum_{i=1}^{N}\sum_{j=1}^{N}|H_{ij}S_{ij}| = \|H \circ S\|_1,$$

where H is a constant matrix with entries $\frac{\rho}{((S_{J-1})_{ij}+\rho)^2}$. It is clear that the entries of H are strictly positive. The symbol "$\circ$" is known as the Hadamard product. The function $\Phi(S)$ defined in (5) is a concave function. However, the surrogate function $\|H \circ S\|_1$ at S_{J-1} is convex, which we refer to as the weighted ℓ_1-norm, where the weights are computed dynamically. We now compare our surrogate function, the 'regular' convex relaxation, and the relaxation suggested in [9] for the case of a single variable in Fig. 2 in the interval $[\rho, 1]$ for a sufficiently small ρ. Our surrogate function for the case of a single variable is given by $\|hx\|_1$, $h = \frac{\rho}{(x+\rho)^2}$; $\rho = 0.005$ is constant.

Figure 2 illustrates that the surrogate is a low-lying flat-like convex in $[\rho, 1]$, allowing an algorithm to iterate over a range and provide sparse solutions using the proximal operator.

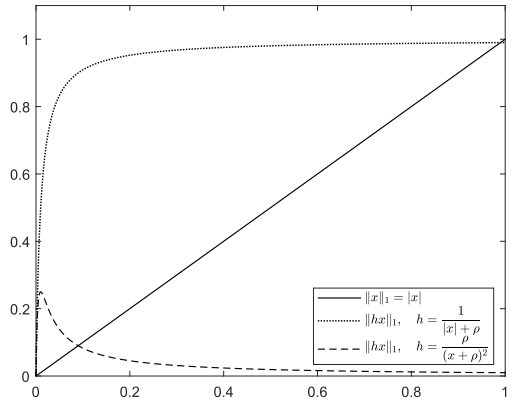


FIGURE 2. Comparison between the convex surrogate function $\|hx\|_1$, the function $\|hx\|_1$, $h = \frac{1}{|x|+\rho}$, $\rho = 0.005$, suggested in [9], and ℓ_1-norm in the case of single variable.

We initialize the optimization algorithm with (L_0, S_0), $H_J = \frac{\rho}{((S_{k-1})_{ij}+\rho)^2} (= H_{k-1})$, $k = 1$, $J = 0$, then update the matrix H after every l number of iterations of the algorithm. We then take the corresponding iterations.

$$\{(L_0, S_0), (L_l, S_l), (L_{2l}, S_{2l}), (L_{3l}, S_{3l}), ...\}, \tag{7}$$

and denote it as the subsequence $\{(L_J, S_J)\}$, where (L_J, S_J) is the last solution at epoch J. The relation between J and k is as follows. If mod $(k, l) = 0$, then we increase J by 1, and update the value of H_J through $H_J = \frac{\rho}{((S_{k-1})_{ij}+\rho)^2}$, where mod (a, b) is the modulo operation that finds the remainder after dividing a by b. Thus, we propose our mathematical model for the MEB problem as follows:

$$\min_{L,S \in \mathbb{R}^{N \times M}} \ \|L\|_* + \lambda \|H \circ S\|_1 \tag{8}$$

$$s.t. \ (2) - (3), \tag{9}$$

where the ℓ_1-norm $\|H \circ S\|_1$ is defined as $\sum_{j=1}^{M} \sum_{i=1}^{N} |H_{ij} S_{ij}|$; the symbol "$\circ$" denotes the Hadamard product. The elements H_{ij} of H are as follows:

$$H_{ij} = \frac{\rho}{(S_{ij} + \rho)^2}, \ \rho > 0, \tag{10}$$

where $H_{ij} > 0$ and ρ is a small constant, e.g. $\rho = 0.05$, where the matrix H (constant matrix with entries $\frac{\rho}{((S_{J-1})_{ij}+\rho)^2}$) is updated at each epoch k of the algorithm used to solve it. We note that at the beginning of each iteration of the algorithm used to solve (8)-(9), the value of S_{ij} will be known. The positive effect of the weight matrix H has been established, both mathematically and numerically.

We have used the ADMM algorithm to solve the above model since the objective function in (8) is separable. Before presenting the algorithm, we start with the singular value thresholding and proximal operator which have been used in all the iterations of ADMM.

Let X be a matrix of size $N \times M$ and of rank r. Assume that the singular value decomposition of X is defined by $X = U\Sigma V^T$, where $U \in \mathbb{R}^{N\times N}$, $V \in \mathbb{R}^{M\times M}$, and $\Sigma \in \mathbb{R}^{N\times M}$. Then for $\delta > 0$, the SVT (Singular Value Thresholding) operator proposed by [7] is given by:

$$SVT_\delta(X) = U\mathcal{D}_\delta(\Sigma)V^T,$$

where $\mathcal{D}_\delta(\Sigma) = Diag(\max\{(\sigma_i - \delta), 0\})$. For a constant matrix $H \in \mathbb{R}^{N\times M}$ and $\delta > 0$, the STT_δ (soft thresholding or shrinkage operator) proposed by [20] is given by:

$$STT_\delta(X, H) = \text{sgn}(X) \circ \max(|X| - \delta H, 0).$$

The scaled form of the augmented Lagrangian is as follows; see [22]:

$$\mathcal{L}_\delta(L, S, \mu) = \|L\|_* + \lambda\|H \circ S\|_1 + \frac{\delta}{2}\|M_b - L - S + \frac{1}{\delta}\mu\|_F^2 - \frac{1}{2\delta}\|\mu\|_F^2, \tag{11}$$

where $\delta > 0$ is the penalty parameter; μ is the Lagrange multiplier; $\|.\|_F$ denotes the Frobenius norm. The entries of the constant matrix are found in the most recent update of S, e.g.; $(H_{k-1})_{ij} = \frac{\rho}{((S_{k-1})_{ij}+\rho)^2}$. The ADMM iterates are as follows:

$$L_k = \arg\min_L \mathcal{L}_\delta(L, S_{k-1}, \mu_{k-1}), \tag{12}$$

$$S_k = \arg\min_S \mathcal{L}_\delta(L_k, S, \mu_{k-1}), \ \textit{and} \tag{13}$$

$$\mu_k = \mu_{k-1} + \delta(M_b - L_k - S_k). \tag{14}$$

The pseudo-code for the ADMM algorithm is given in **Algorithm** 1.

Algorithm 1 The ADMM algorithm for finding the maximum bi-clique

Input: Adjacency matrix M_b, regularization parameter $\lambda > 0$, $\rho > 0$, and $\delta > 0$

Initialization: Start with feasible $LJ = L_{k-1}$, $S_J = S_{k-1}$, $H_{k-1} = H(= H_J)$, and set $\mu_{k-1} = 0$, $k = 1$, $J = 0$

(a) Set $(H_{k-1})_{ij} = \frac{\rho}{((S_{k-1})_{ij}+\rho)^2}$, for $i \in \{1, 2, ..., N\}$, $j \in \{1, 2, ..., M\}$, where $(S_{k-1})_{ij} \in \{0, 1\}$

WHILE Stopping condition not satisfied

(b)

$$\begin{aligned} L_k &= \arg\min_L \ \frac{1}{\delta}\|L\|_* + \frac{1}{2}\|M - L - S_{k-1} + \frac{1}{\delta}\mu_{k-1}\|_F^2 \\ &= SVT_{\frac{1}{\delta}}\left(M_b - S_{k-1} + \frac{1}{\delta}\mu_{k-1}\right) \end{aligned}$$

(c)

$$\begin{aligned} S_k &= \arg\min_S \ \frac{\lambda}{\delta}\|H \circ S\|_1 + \frac{1}{2}\|M - L_k - S - \frac{1}{\delta}\mu_{k-1}\|_F^2 \\ &= STT_{\frac{\lambda}{\delta}}\left(M_b - L_k + \frac{1}{\delta}\mu_{k-1}, H_{k-1}\right) \\ &= \text{sgn}\left(M_b - L_k + \frac{1}{\delta}\mu_{k-1}\right) \circ \max\left(\left|M_b - L_k + \frac{1}{\delta}\mu_{k-1}\right| - \delta H_{k-1}, 0\right) \end{aligned} \tag{15}$$

(d) Update μ_k via (14) and $(H_k)_{ij} = \frac{\rho}{((S_k)_{ij}+\rho)^2}$, set $k = k + 1$

(e) If $\mod(k, l) = 0$, then update $(H_k)_{ij} = \frac{\rho}{((S_k)_{ij}+\rho)^2}$, set $L_J = L_k$, $S_J = S_k$, $H_J = H_k$, $\mu_J = \mu_k$, and $J = J + 1$.

ENDWHILE

Output: The recovered matrix L_k represents the bi-clique

Remark 3.1 (Effect of Initialization)**.** The performance of Algorithm 1 is influenced by the choice of initialization for the matrices L and S. In our experiments, two strategies were compared:

- **Random initialization:** S_0 is initialized with 1s with probability p, and $L_0 = M_b - S_0$. *This initialization strategy is adopted throughout this paper.*
- **Zero initialization:** $S_0 = 0$, and $L_0 = M_b$.

The algorithm converged more quickly with **random initialization**, requiring fewer iterations and exhibiting slightly better numerical accuracy. However, both strategies led to successful recovery of the bi-clique, confirming the robustness of the algorithm with respect to initialization.

From steps (a) and (d) in **Algorithm** 1, we see that the weights used in $H \circ S$ change with each iteration k. These adaptive weights are central to obtaining integer solutions, which are explained below.

The 'regular' model (1)-(3) can be solved by adapting **Algorithm** 1 where the ADMM iterates are as follows:

$$L_k = \text{SVT}_{\frac{1}{\delta}}\left(M - S_{k-1} + \frac{1}{\delta}\mu_{k-1}\right), \quad \text{and} \tag{16}$$

$$\begin{aligned} S_k &= \text{STT}_{\frac{\lambda}{\delta}}\left(M - L_k + \frac{1}{\delta}\mu_{k-1}, I\right) \\ &= \text{sgn}\left(\left[M - L_k + \frac{1}{\delta}\mu_{k-1}\right]_{ij}\right) \circ \max\left(\left|\left[M - L_k + \frac{1}{\delta}\mu_{k-1}\right]_{ij}\right| - \frac{\lambda}{\delta}, 0\right). \end{aligned} \tag{17}$$

where I is the identity matrix. The update of μ_k is the same as in (14).

We compare $\lambda\|H \circ S\|_1$ in our model (8)-(9) with $\lambda\|S\|_1$ in the 'regular' matrix decomposition model (1)-(3), using the iterates of **Algorithm** 1. A comparison of the ADMM iterates (15) and (17) of S_k shows that (15) carries additional information from $(k-1)$-th to k-th iteration via $(H_{k-1})_{ij} = \frac{\rho}{((S_{k-1})_{ij}+\rho)^2}$. It follows from H_{k-1} that $(S_{k-1})_{ij} \longrightarrow 0 \Rightarrow (H_{k-1})_{ij} \longrightarrow 1/\rho$, $\rho << 1$.

Before making further comparisons, we look at the shrinkage parameter $\tau = \frac{\lambda}{\delta}$ in (15) and (17). The theoretical value $\lambda = \frac{1}{\sqrt{\max\{N,M\}}}$ has been suggested by [8]. Clearly, $\tau = \frac{\lambda}{\delta}$ is a small fraction provided $\delta > 1$ (a value we have implemented). A comparison of the shrinkage operators (15) and (17) suggests that if $(S_{k-1})_{ij}$ is small or close to zero, then $(\frac{\lambda}{\delta})(H_{k-1})_{ij}$ in (15) is larger than $\frac{\lambda}{\delta}$ in (17). This implies that

$$(S_k)_{ij} = \text{sgn}\left(\left[M - L_k + \frac{1}{\delta}\mu_{k-1}\right]_{ij}\right) \circ \max\left(\left|\left[M - L_k + \frac{1}{\delta}\mu_{k-1}\right]_{ij}\right| - \frac{\lambda}{\delta}, 0\right),$$

has more chance of staying fractional than

$$(S_k)_{ij} = \text{sgn}\left(\left[M - L_k + \tfrac{1}{\delta}\mu_{k-1}\right]_{ij}\right) \circ \max\left(\left|\left[M - L_k + \tfrac{1}{\delta}\mu_{k-1}\right]_{ij}\right| - \tfrac{\lambda}{\delta}(H_{k-1})_{ij}, 0\right),$$

as $\frac{\lambda}{\delta}(H_{k-1})_{ij} > \frac{\lambda}{\delta}$. In later stages of the algorithm, when the (majority) entries of S_{k-1} approach zero in iteration $k-1$ then this information is fed into iteration k via H_{k-1} with $(H_{k-1})_{ij} > 1$, $(H_{k-1})_{ij} \longrightarrow \frac{1}{\rho}$ when $(S_{k-1})_{ij} \longrightarrow 0$. This increases the probability that $\left(\left|\left[M - L_k + \frac{1}{\delta}\mu_{k-1}\right]_{ij}\right| - \frac{\lambda}{\delta}(H_{k-1})_{ij}\right)$ in (15) is negative, resulting in $(S_k)_{ij} = 0$. The iterate (17) of the regular model lacks this characteristic,

therefore $(S_k)_{ij}$ remains a fraction if $\left|\left[M - L_k + \frac{1}{\delta}\mu_{k-1}\right]_{ij}\right| > \frac{\lambda}{\delta}$. On the other hand, $(S_{k-1})_{ij}$ approaching 1 implies $(H_{k-1})_{ij} < 1$. However, in this case, the integer value of $(S_k)_{ij}$ is a gradual optimization process rather than an immediate event.

4. **Theoretical guarantee for exact recovery.** Let the rank of L be r. The low-rank matrix L can be written as $L = U\Sigma V^T$, $U = [U_1, U_2, ..., U_r]$, $V = [V_1, V_2, ..., V_r]$, where U and V are the left and the right singular vectors of L, respectively; Σ is a diagonal matrix with diagonal entries $(\sigma_1, \sigma_2, ..., \sigma_r)$, where σ_i is the i-th singular value of L.

Let $\mathcal{T}$ denote the linear space of low-rank matrices; see [8]:

$$\mathcal{T} := \{UZ^\top + YV^\top \mid Z \in \mathbb{R}^{N\times r},\ Y \in \mathbb{R}^{M\times r}\}, \tag{18}$$

and denote by Ψ the linear space of sparse matrices, see [8]:

$$\Psi := \{S \in \mathbb{R}^{N\times M} \mid |supp(S)| = s\}. \tag{19}$$

Based on some assumptions about low-rank and sparse matrices, we present the incoherence conditions for the guaranteed recovery for L and S. Let us start with a rank-one matrix L_*, the optimal solution to (8)-(9). We can easily see that the only non-zero singular value (the maximum singular value) of L_* is $\sqrt{nm}$. That is $\sigma_1 = \|L_*\| = \|L_*\|_* = \sqrt{\sum_{i=1}^{\min(N,M)} \sigma_i^2} = \|L_*\|_F = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M}(L_*)_{ij}^2} = \sqrt{nm}$, since $\sigma_i = 0,\ i = 2, 3, ..., \min(N, M)$. It follows that the elements of $UU^\top$ and $VV^\top$ are from the sets $\{0, \frac{1}{n}\}$ and $\{0, \frac{1}{m}\}$, respectively, and the elements of U and V are from the sets $\{0, \frac{1}{\sqrt{n}}\}$ and $\{0, \frac{1}{\sqrt{m}}\}$, respectively, since $L_* = \sqrt{nm}UV^\top$.

For the perfect disentanglement of the low-rank and the sparse components, we need to assume that the low-rank component L_* is not sparse. This can be accomplished by ensuring that the columns space of L_* is not aligned with the standard basis vectors. This is guaranteed by the following conditions in [8] with a parameter μ:

$$\max_j \|U^\top e_j\|_2^2 \le \frac{\mu r}{N},\ \max_j \|V^\top e_j\|_2^2 \le \frac{\mu r}{M},\ \text{with } 1 \le \mu \le \frac{\min(N, M)}{r},$$

and

$$\|UV^\top\|_\infty \le \sqrt{\frac{\mu r}{NM}},\ \text{with } 1 \le \mu \le \frac{\sqrt{NM}}{r}.$$

Here, $\|.\|_2$ is the Euclidean norm, and $\|UV^\top\|_\infty$ is the infinity norm defined as the maximum component of $UV^\top$ in magnitude.

For our problem, it is easy to see that

$$\frac{1}{n} = \max_j \|U^\top e_j\|_2^2 \le \frac{\mu_0 r}{N},\ \frac{1}{m} = \max_j \|V^\top e_j\|_2^2 \le \frac{\mu_1 r}{M}, \forall j, \tag{20}$$

hold for $1 < \mu_0 \le \frac{N}{r}$ and $1 < \mu_1 \le \frac{M}{r}$, respectively. By construction of M_b and L_*, $UV^\top$ has nm non-zero entries, each of the entries is $\frac{1}{\sqrt{nm}}$. It is also easy to see for our problem that the joint incoherent condition holds when $1 < \mu \le \frac{\sqrt{NM}}{r}$ since

$$\frac{1}{\sqrt{nm}} = \|UV^\top\|_\infty \le \sqrt{\frac{\mu r}{NM}},\ 1 < \mu \le \frac{\sqrt{NM}}{r}. \tag{21}$$

The equations in (20) and (21) show that both incoherent conditions hold for our matrix L_*. The second condition for guaranteed recovery is that the sparsity pattern of S_* does not have to be too structured. To achieve this; Bernoulli model with probability p will be considered. We construct S_* such that $|supp(S_*)| < p(NM - nm)$, this due to the fact that $n \times m$ entries out of $N \times M$ are fixed to be ones for the bi-clique; in this case, we can adjust the probability p for the support of S_*. Thus, we use probability p to construct S_* such that $(S_{ij})_* = 1$ with probability p and 0 with probability $1 - p$. To determine the structure of the sparsity pattern in S_*, the variances of each of the columns or rows are calculated. The components of each row are associated with a random variable that assumes 1 with probability p and 0 with probability $1 - p$. Therefore, it follows that the mean and variance of the random variables are p and $p(1 - p)$. For each row and column, the expected cardinality is Np and Mp, respectively. Similarly, the variance for each row and column is $Np(1 - p)$ and $Mp(1 - p)$, respectively. Therefore, we can conclude that no pattern is guaranteed, since the expected value of $Np(1 - p)$ or $Mp(1 - p)$ is the same for every row and column. That is,

$$\left|Var(S_{i*}) - Var(S_{j*})\right| < \varepsilon_1, \ \left|Var(S_{*i}) - Var(S_{*j})\right| < \varepsilon_2, \tag{22}$$

for any $\varepsilon_1, \varepsilon_2 > 0$, where $Var(S_{i*})$ and $Var(S_{j*})$ are the variances of the entries of the i-th row and the j-th column, respectively.

Given the above incoherence conditions (20)-(22) in L_* and S_*, recovery is guaranteed by convex optimization. We have the following theorem.

Theorem 4.1. *Suppose L_* is an $N \times M$ matrix of rank r that obeys the incoherence conditions (20) and (21). Moreover, the entries for all rows or columns of S_* satisfy (22). Then there exists a numerical constant c such that, with probability at least $1 - cN^{-10}$, the output (L_*, S_*) of the optimization problem*

$$\min_{L,S} \ \|L\|_* + \lambda \|H \circ S\|_1, \tag{23}$$

$$s.t. \ M_b = L + S,$$

$$S_{ij} \in [0, 1], \tag{24}$$

$\lambda = \frac{\alpha}{\sqrt{\max(N,M)}}$, $0.0027 < \alpha < 0.15$, *is exact, provided that*

$$rank(L_*) \leq \tau \sqrt{\frac{N}{\rho_0 \log^2 \max(N, M)}}, \ \tau \geq 1, \ s < NM - nm,$$

$\rho_0 = \frac{s}{NM}$, *where $s < NM - nm$, is a positive constant.*

The proof of Theorem 4.1 is based on the assumptions of specific standard identifiability conditions for the sparse and low-rank components in (20), (21), and (22), see [8, 22]. The range of α has been numerically found in the section below.

5. **Computational results.** In this section, we present computational results of our algorithm on the planted bi-clique problems, MEB from randomly generated graphs, and on a number of real-world bi-clique problems. All experiments are computed in Matlab 2021b, under the following machine configurations: Intel core i7, 3.60GHz CPU and 16GB of RAM. Here, we evaluate the performance of **Algorithm 1** by applying it to identify the planted bi-cliques in given graphs, to find maximum bi-cliques in random graphs (where no bi-cliques are planted) and finally to identify bi-cliques for real-world graphs.

Parameter selection and sensitivity. The performance of **Algorithm** 1 depends on a few critical parameters: the regularization parameter λ, the weight parameter ρ, and the penalty parameter δ. In the following, we summarize the selection strategy and sensitivity of each.

Regularization parameter λ: Following the theoretical results in [8], we use $\lambda = \frac{\alpha}{\sqrt{\max(N,M)}}$, where α is a scalar that controls the trade-off between low-rank and sparsity. Through numerical trials, we found that $\alpha \in [0.0027, 0.15]$ works well in various graph sizes. For all experiments in this paper, we use $\alpha = 0.08$, which yields a stable and near-optimal performance. Sensitivity tests show that performance is only slightly outside this range.

Weight parameter ρ: This parameter controls the shape of the weighting function $H_{ij} = \frac{\rho}{(S_{ij}+\rho)^2}$. A small ρ emphasizes zero entries in S more strongly, encouraging sparsity. We fix $\rho = 0.05$ based on previous studies [22] and confirm its effectiveness in datasets. Values smaller than 0.01 may over-penalize and hinder convergence.

Penalty parameter δ: The augmented Lagrangian penalty δ affects the rate of convergence in ADMM. We use a data-adaptive setting $\delta = \frac{1}{\text{mean}(M_b)}$, where $\text{mean}(M_b)$ is the average value in the adjacency matrix. Empirical results show that this choice balances the update scales in L, S, and the multiplier μ.

In general, the algorithm shows strong robustness to parameter variations, particularly when λ and ρ are kept within the above ranges.

5.1. **Planted bi-cliques.** Let $V_1^* \times V_2^*$ denote the planted bi-clique, $|V_1^*| = n$, $|V_2^*| = m$. Let M_b represent the adjacency matrix of the bipartite graph $G_b(V_1, V_2, E)$, $|V_1| = N$, and $|V_2| = M$. We set $(M_b)_{ij} = 1$ for $(i,j) \in V_1^* \times V_2^*$. The remaining edges $(i,j) \in (V_1 \times V_2) \backslash (V_1^* \times V_2^*)$ are included with probability p. The proposed algorithm has been tested and achieved a very similar accuracy for $p \in [0.5, 0.85]$ for all the problems considered. However, here we present results based on a value of p of 0.5.

Algorithm 1 has been initialized with a feasible S, generated randomly, of ones with probability $p = 0.25$ and zeros with probability $1 - p = 0.75$. The feasible L is then initialized as $L = M_b - S$. Throughout our numerical tests, we have used a constant regularization parameter λ. According to our numerical analysis, **Algorithm** 1 provides almost insensitive results for $\lambda = \frac{\alpha}{\sqrt{\max(N,M)}}$ for any $\alpha \in [0.0027, 0.15]$, see [22]. We have estimated the range, $[l, u]$, for α as follows. First, we calculate three ranges $[l_i, u_i]$, $i = 1, 2, 3$, corresponding to $c = 0.25, 0.5$ *and* 0.75, respectively, in $s = \frac{1}{2}(NM - nm)$, $n = cN$, $m = cM$. We plot α $(= \max(N,M)/s)$ against $\max(N,M)$ for each value of c and obtain $[l_i, u_i]$ for α. We then take $l = \min l_i$ and $u = \max u_i$, $i = 1, 2, 3$. We have used $\alpha = 0.08$ for all pairs (N, n) and (M, m) for the results presented here. The values of ρ in (10) are within the range $[0.05, 0.42]$, see [22]. We have used $\rho = 0.05$ for the results presented here. The value of δ in (11) we have used is $\delta = \frac{1}{\text{mean}(M_b)}$, where $\text{mean}(M_b)$ is the average value of the entries in M_b.

For the regular model, the final solution of the ADMM algorithm is denoted (L^1, S^1), while for the proposed model (8)-(9), the final solution is denoted (L^2, S^2).

For each algorithm, the relative error $errL^i$ is calculated using the Frobenius norm.

$$errL^i = \frac{\|L^i - L_*\|_F}{\|L_*\|_F}, \ i = 1, 2,$$

where L_* corresponds to $V_1^* \times V_2^*$, the planted bi-clique.

We terminate **Algorithm** 1 when

$$\|M_b - L_k - S_k\|_F \leq \varrho \tag{25}$$

holds, where (L_k, S_k) is the solution in iteration k, $\varrho = 0.0001$.

We have used bipartite graphs of sizes 200×150, 500×350, and 1000×800. For a bi-clique of size $n \times m$, we fixed n and set $m = 10, 20, ..., M - 10$, fix the value of m and let $n = 10, 20, ..., N - 10$. For each value of n and m the procedure described above to generate the bipartite graph M_b is repeated 15 times. Hence, the number of problems considered for graphs of sizes 200×150, 500×350, and 1000×800 is 495, 1245, and 2535, respectively, and therefore the total number of test runs was 4275.

Convergence analysis. The convergence of **Algorithm** 1 was observed across all test cases by monitoring the residual $\|M_b - L_k - S_k\|_F$. We set the stopping criterion to $\varrho = 10^{-4}$, and in all experiments, the algorithm converged within 50–120 iterations depending on the size of the problem. The residual decay was smooth and monotonic, confirming the stability of the ADMM updates. No divergence or oscillatory behavior was detected in any of the runs, indicating robust numerical performance across both planted and random graph instances.

We first start with a bipartite graph of size 200×150 where the input adjacency matrix M_b is constructed as above. For the bi-clique of size $n \times m$, we fix the value of n to 100 and vary the value of m; we have used $m = 10, 20, ..., M - 10$. We also fix the value of m to be 70 and vary the value of n; we have used $n = 10, 20, ..., N - 10$.

Comparison of methods. To validate the effectiveness of our proposed model, we compare it with two baseline methods: the 'regular' matrix decomposition model and the Densest Sub-graph Algorithm (DSA). The 'regular' model refers to the classical convex relaxation based on nuclear norm and ℓ_1-norm minimization, formulated as:

$$\min_{L,S} \ \|L\|_* + \lambda \|S\|_1 \quad \text{subject to} \quad M_b = L + S, \quad S_{ij} \in [0, 1],$$

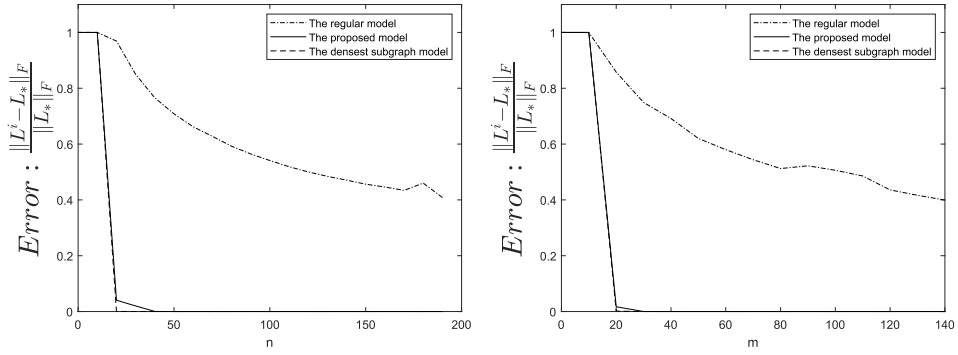
which we solve using the same ADMM framework as **Algorithm** 1 but with fixed weights (that is, $H = \mathbf{I}$, the identity matrix). This model does not dynamically adapt to the sparsity structure of S and often results in fractional solutions with a lower sparsity.

The DSA method, introduced in [6], formulates the densest sub-graph problem as a convex optimization problem and requires the densest sub-graph size (bi-clique size, which is $n \times m$) as input. This limits its applicability to cases where the true size of the planted or maximum bi-clique is known in advance. We implemented DSA according to the authors' specifications and used known planted sizes for a fair comparison.

Together, these two baselines allow us to evaluate the advantages of our adaptive weighted ℓ_1-based model in terms of accuracy, convergence, and robustness to initialization and noise.

We have compared **Algorithm** 1 with the Densest Subgraph Algorithm (DSA) proposed by [6] for all the problems considered in this section. Before comparisons are made between our algorithm and DSA, it is important to note that the implementation of DSA has an intrinsic requirement. Unlike our algorithm, DSA requires the size $n \times m$ of the planted bi-clique as input, and therefore cannot be applied

to randomly generated graphs. Figure 3 shows the average errors obtained, where the y axis indicates the average of relative errors; this average is determined over 15 runs for each problem. The value n or m on the x-axis denotes the varied part of the size of the planted bi-clique while the other part remains fixed.



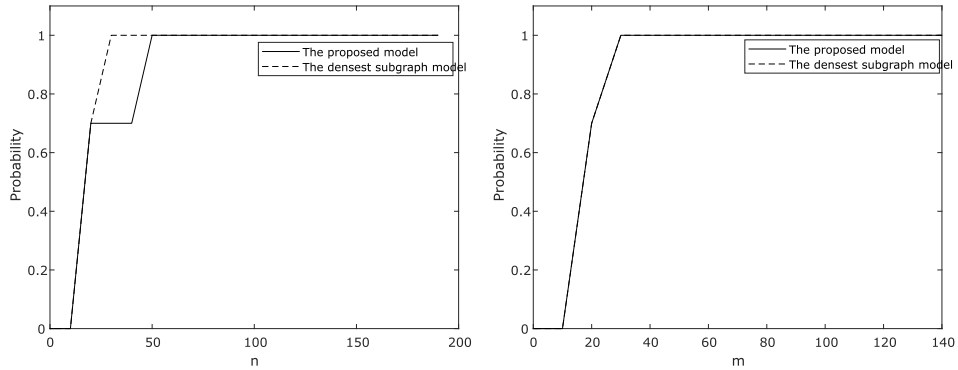
(A) $N = 200, M = 150$, and with $m = 70$ fixed

(B) $N = 200, M = 150$, and with $n = 100$ fixed

FIGURE 3. The average recovery errors for bi-clique of size nm, $M_b \in \mathbb{R}^{200\times 150}$.

Figure 3 shows that the ADMM algorithm for the regular model has worse errors than that of the proposed model for all pairs (N, M) and (n, m). In addition, the ADMM algorithm for the regular model shows that the error improves for higher values of n and m, that is, for the easiest problems. On the other hand, our proposed model (8)-(9) achieves errors less than 10^{-8} for all $n, m \geq 30$ for $N = 200$. However, the errors produced by DSA are about 10^{-5}.

We now present the probability of recovery for each problem where recovery means that the obtained solution has an average error less than 10^{-8} for **Algorithm 1** and about 10^{-5} for DSA, see [6], DSA fails to produce errors less than 10^{-5}. This is demonstrated in Figure 4 for all N, M, n and m. Figure 4 shows that both algorithms are comparable, noting that ADMM produces smaller errors than DSA.



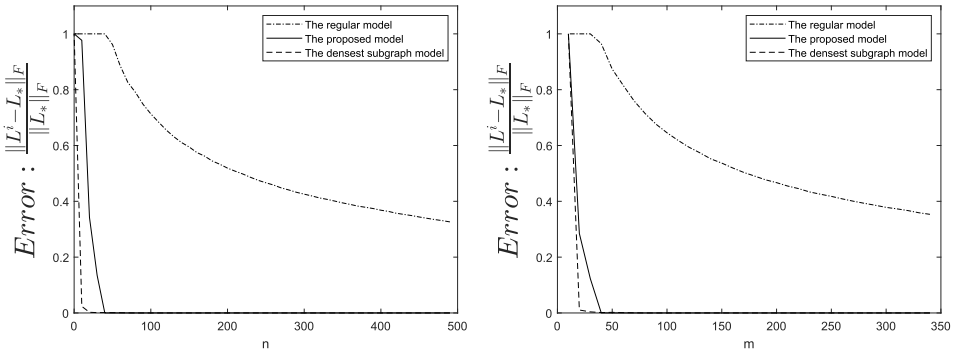
(A) $N = 200, M = 150$, and with $m = 70$ fixed

(B) $N = 200, M = 150$, and with $n = 100$ fixed

FIGURE 4. The average recovery probabilities for bi-clique of size nm, $M_b \in \mathbb{R}^{200\times 150}$.

The average recovery probabilities for bi-cliques of sizes $n \times m$ are presented in Figure 4a (respectively, Figure 4b) correspond to the results in Figure 3a (respectively, 3b).

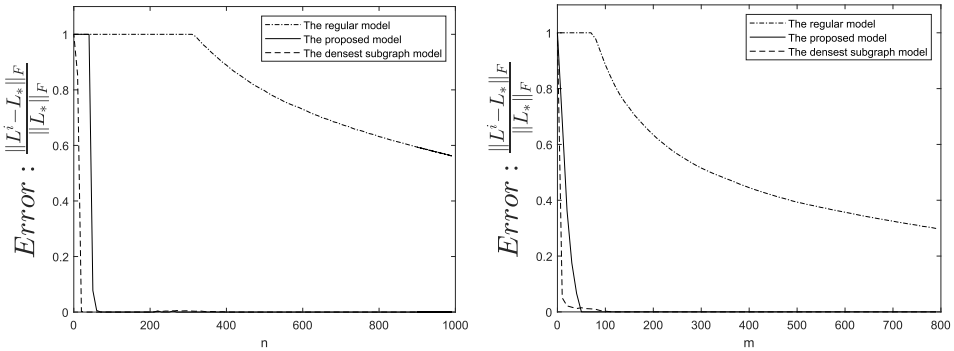
We now perform numerical studies using larger values for N, M, n, and m. The relative error values are shown in Figure 5 and Figure 6 for problems corresponding to $M_b \in \mathbb{R}^{500\times 350}$ and $M_b \in \mathbb{R}^{1000\times 800}$, respectively.



(A) $N = 500, M = 350$, and with $m = 200$ fixed

(B) $N = 500, M = 350$, and with $n = 250$ fixed

FIGURE 5. The average recovery errors for bi-clique of size nm, $M_b \in \mathbb{R}^{500\times 350}$.



(A) $N = 1000, M = 800$, and with $m = 150$ fixed

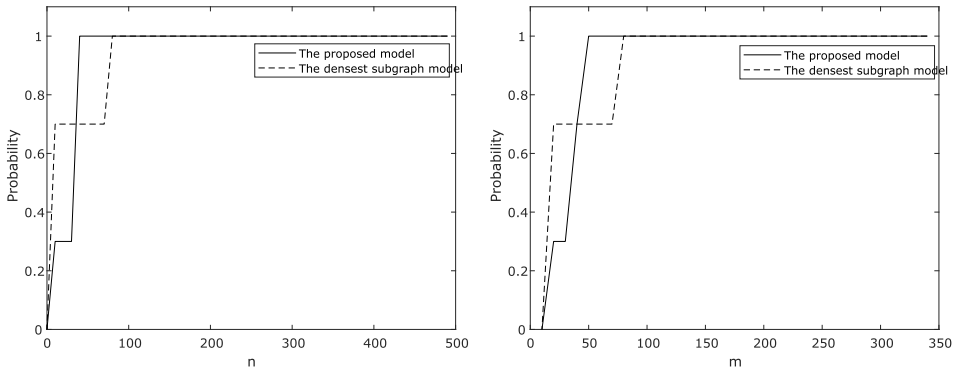
(B) $N = 1000, M = 850$, and with $n = 600$ fixed

FIGURE 6. The average recovery errors for bi-clique of size nm, $M_b \in \mathbb{R}^{1000\times 800}$.

Figures 5 and 6 show that, while ADMM produces small errors than DSA; both algorithms outperform the regular model.

The average recovery probabilities for bi-cliques of size nm using $M_b \in \mathbb{R}^{500\times 350}$ and $M_b \in \mathbb{R}^{1000\times 800}$ are shown in Figures 7 and 8, respectively. Figure 5a (respectively, Figure 5b) corresponds to the results in Figure 7a (respectively, 7b).

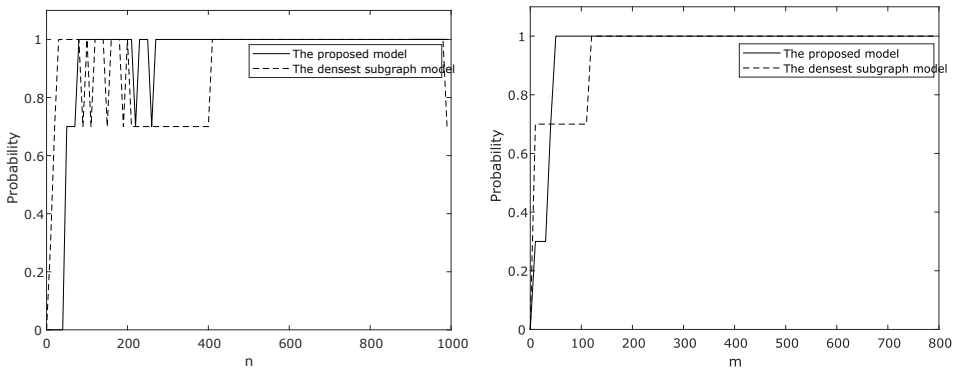
Similarly, Figure 6a (respectively Figure 6b) corresponds to the results in Figure 8a (respectively 8b).



(A) $N = 500, M = 350$, and with $m = 200$ fixed

(B) $N = 500, M = 350$, and with $n = 250$ fixed

FIGURE 7. The average recovery probabilities for bi-clique of size nm, $M_b \in \mathbb{R}^{500\times350}$.



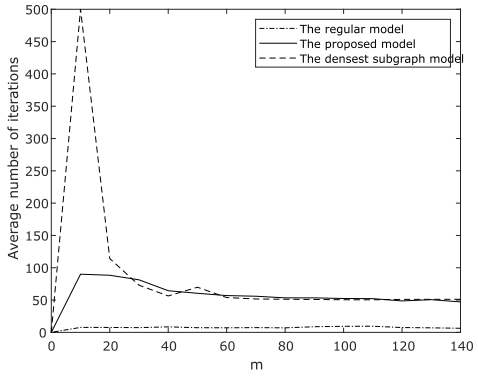
(A) $N = 1000, M = 800$, and with $m = 150$ fixed

(B) $N = 1000, M = 800$, and with $n = 600$ fixed

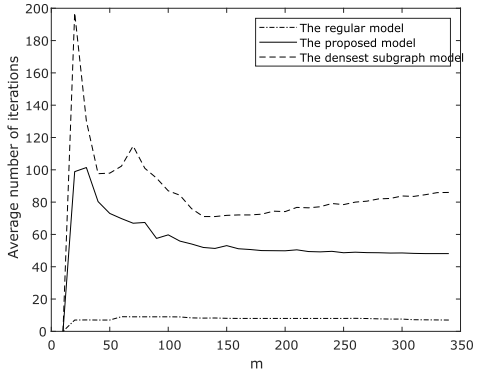
FIGURE 8. The average recovery probabilities for bi-clique of size nm, $M_b \in \mathbb{R}^{1000\times800}$.

The results summarized in Figures 7 and 8 demonstrate near-perfect recovery of planted bi-cliques with our algorithm with probability equal to one for most of the cases considered. However, DSA does not achieve perfect recovery for several problems as shown in Figures 7 and 8.

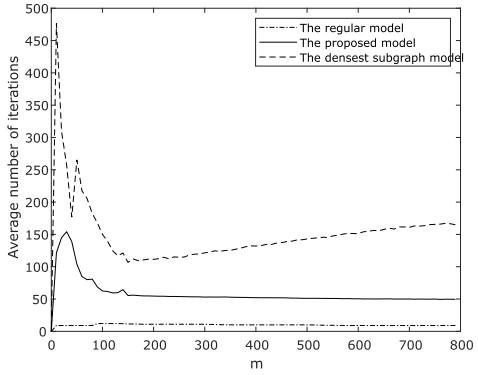
In Figure 9 we present the average number of iterations needed by **Algorithm** 1 for producing average error of 10^{-8} and DSA to produce an average error of 10^{-5}.



(A) $N = 200$, $M = 150$, and with $n = 100$ fixed

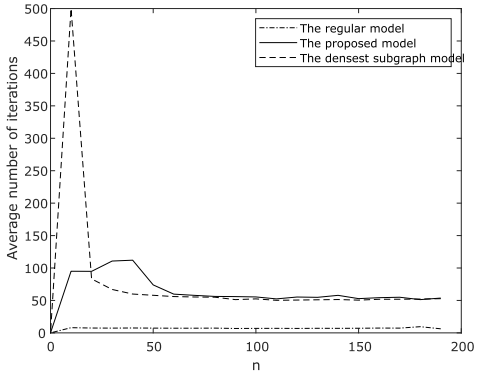


(B) $N = 500$, $M = 350$, and with $n = 250$ fixed

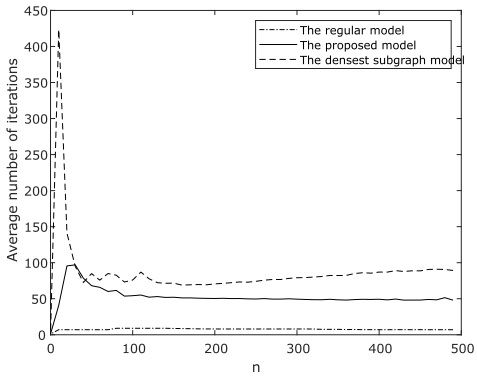


(C) $N = 1000$, $M = 800$, and with $n = 600$ fixed

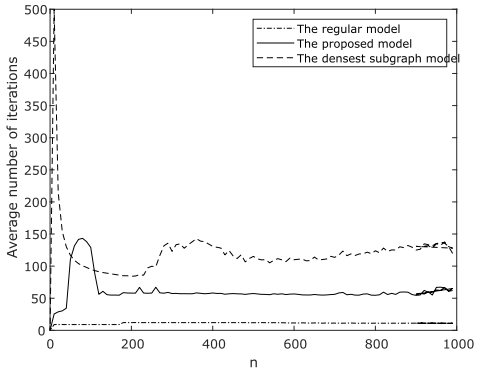
FIGURE 9. The average number of iterations for each problem with varied m.



(A) $N = 200$, $M = 150$, and with $m = 70$ fixed



(B) $N = 500$, $M = 350$, and with $m = 200$ fixed

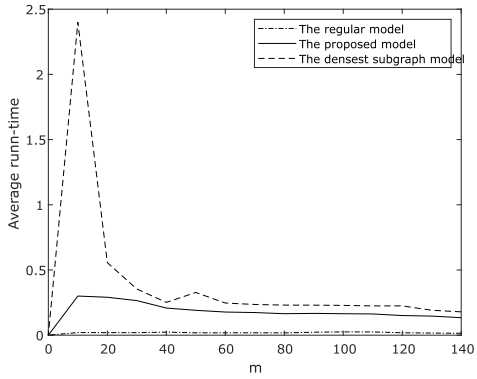


(C) $N = 1000$, $M = 800$, and with $m = 150$ fixed

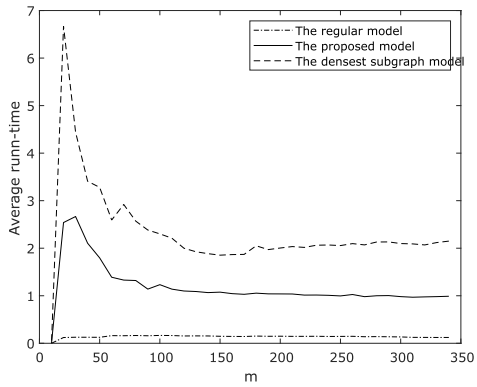
FIGURE 10. The average number of iterations for each problem with varied n.

Figures 9 and 10 show that our algorithm produces better quality errors with a smaller number of iterations than DSA on average.

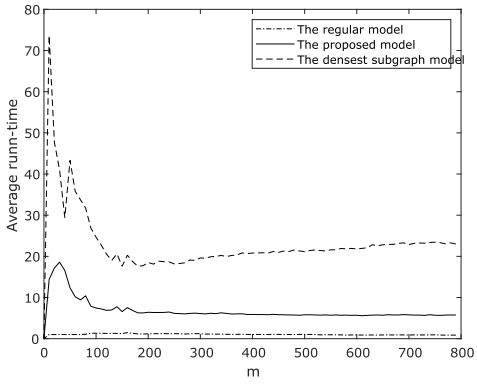
The total run-time and average run-time per iteration needed by our algorithm and DSA are summarized in the following figures.



(A) $N = 200,\ M = 150$, and with $n = 100$ fixed

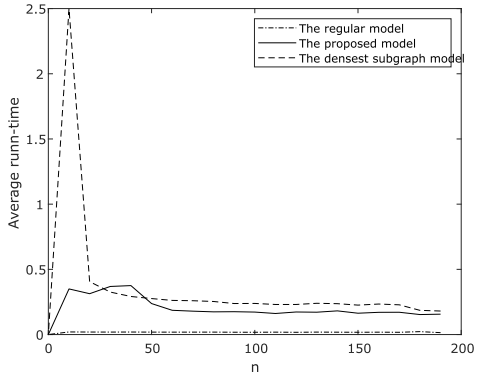


(B) $N = 500,\ M = 350$, and with $n = 250$ fixed

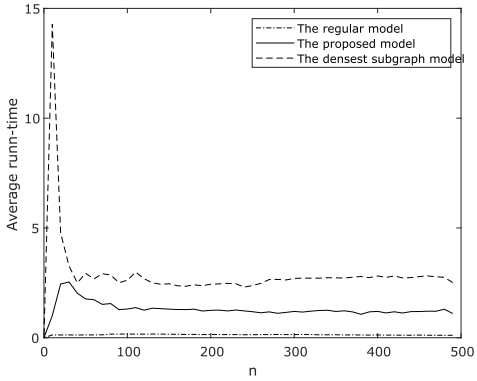


(C) $N = 1000,\ M = 800$, and with $n = 600$ fixed

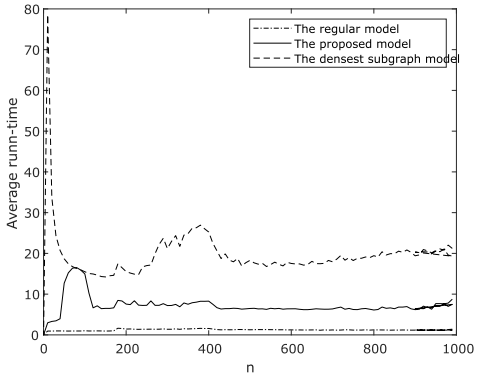
FIGURE 11. Average runtime for each problem with varied m.



(A) $N = 200$, $M = 150$, and with $m = 70$ fixed



(B) $N = 500$, $M = 350$, and with $m = 200$ fixed

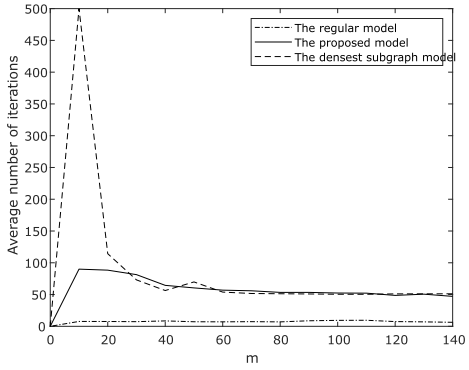


(C) $N = 1000$, $M = 800$, and with $m = 150$ fixed

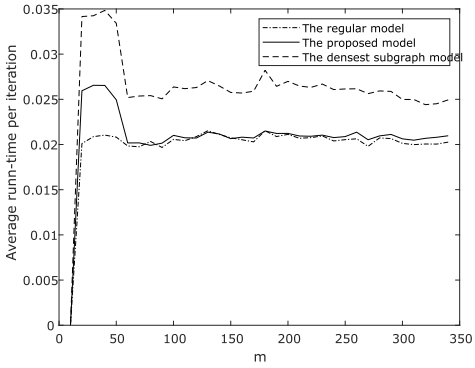
FIGURE 12. Average runtime for each problem with varied n.

Figures 11 and 12 show the average run-time for all pairs (N, n) and (M, m) for the problems considered. The averages are taken over 15 runs on each problem. These figures show that our algorithm performs better than DSA in finding the optimal solution in both cases when m and n are varied.

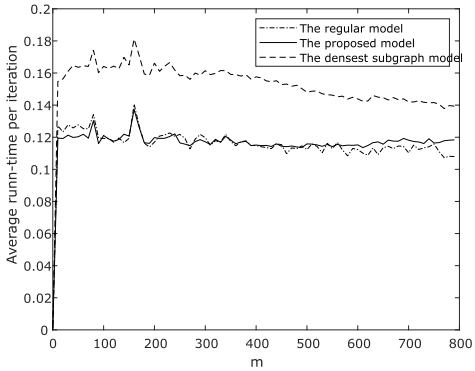
Figures 13 and 14 show the average runtime per iteration for all the problems considered, where our algorithm performs slightly better, on average.



(A) $N = 200$, $M = 150$, and with $n = 100$ fixed

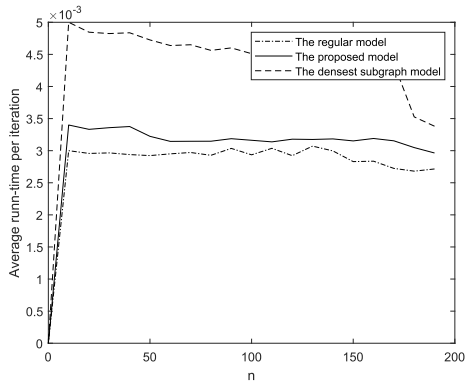


(B) $N = 500$, $M = 350$, and with $n = 250$ fixed

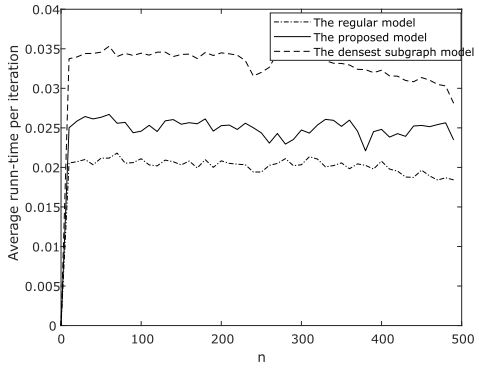


(C) $N = 1000$, $M = 800$, and with $n = 600$ fixed

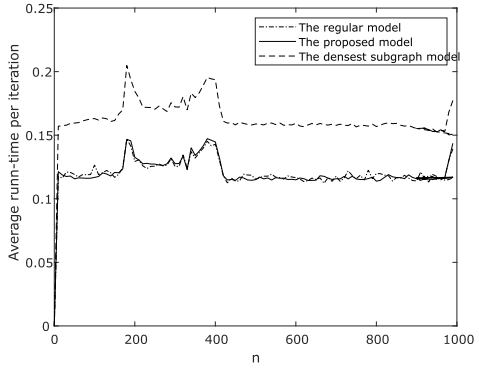
FIGURE 13. Average runtime per iteration for each problem with varied m.



(A) $N = 200$, $M = 150$, and with $m = 70$ fixed



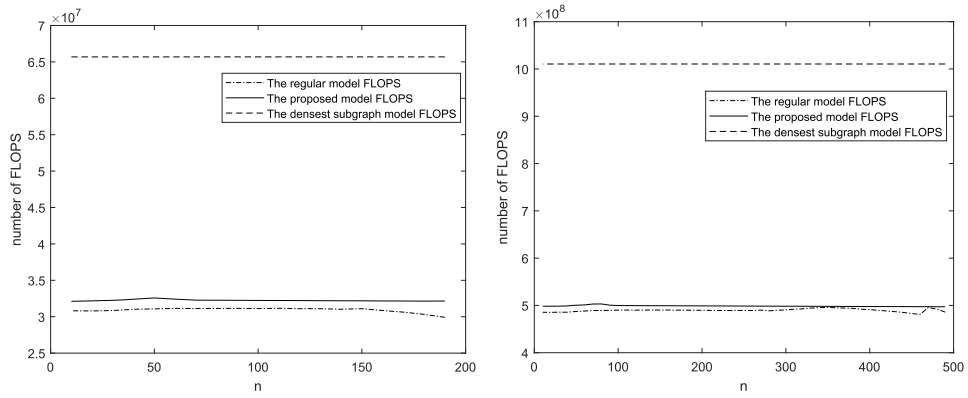
(B) $N = 500$, $M = 350$, and with $m = 200$ fixed



(C) $N = 1000$, $M = 800$, and with $m = 150$ fixed

FIGURE 14. Average runtime per iteration for each problem with varied n.

To clarify the scaling of our proposed approach, we sketch the number of FLOPS (Floating Point Operations per Second) needed per iteration. Figure 15 shows the average number of FLOPS needed per iteration for $N = 200$ and $N = 500$.



(A) $N = 200$, $M = 150$, and with $m = 70$ fixed

(B) $N = 500$, $M = 350$, and with $m = 200$ fixed

FIGURE 15. Average number of FLOPS per iteration.

Figure 15 shows that the number of FLOPS needed per iteration is $\mathcal{O}(N^3)$. It also shows that our proposed algorithm needed less number of FLOPS than DSA.

5.2. **Bi-cliques in random bipartite graphs.** We have also performed experiments on random bipartite graphs in which edges are assigned with probability p. These results are presented in the following table, where $n \times m$ is the size of the maximum bi-clique obtained by our algorithm. We have used the same stopping condition in (25) but calculated the errors using the condition.

$$Error = \left| \sum_{i=1}^{N} \sum_{j=1}^{M} (L_*)_{ij} - \|L_*\| \right|, \tag{26}$$

where L_* is the optimal low-rank solution obtained. For random graphs, it is difficult to accurately estimate the errors of the solved problem. However, due to our established result $\|L_*\| = \sigma_1 = \sqrt{mn}$, shown earlier, our suggested formula (26) serves as a natural measure for the recovery of the maximum bi-clique based on the nodes in the bi-clique.

The solution (L, S) is initialized as in the case of planted bi-clique. A total of 48 runs have been performed considering a number of (N, M) pairs. The results presented in each row of Table 3 are obtained from a single run. We have used two values for N, that is, $N = 200, 500$ and varied M to create the pair (N, M) for this experiment. We have used $p = 0.85$ to generate $M_b \in \mathbb{R}^{200\times M}$ and $p = 0.88$ for $M_b \in \mathbb{R}^{500\times M}$ for each M considered. We have used higher values for p to ascertain that a reasonable size of the maximum bi-clique, (n, m), is formed in each random graph. The results obtained show the perfect recovery of maximum bi-cliques that have little or no errors.

Four random bipartite graphs are generated for each pair (N, M). The corresponding recovered bi-clique sizes are shown in the column under (n, m).

TABLE 1. Random graph with $N = 200$ fixed

(N, M)	(n, m)	CPU time	Error
$(200, 170)$	$(133, 119)$	1.7462	0
	$(113, 119)$	1.7383	$3.9790e^{-13}$
	$(124, 104)$	1.6812	$4.2633e^{-14}$
	$(111, 106)$	1.6811	$7.1054e^{-14}$
$(200, 145)$	$(111, 18)$	1.2597	$2.8422e^{-14}$
	$(132, 107)$	1.5079	$1.4211e^{-14}$
	$(108, 103)$	1.4105	$1.4211e^{-14}$
	$(101, 101)$	1.3989	$2.844e^{-14}$
$(200, 100)$	$(139, 82)$	0.7479	$2.2737e^{-13}$
	$(126, 91)$	0.8779	$9.9476e^{-14}$
	$(118, 70)$	0.7656	0
	$(95, 86)$	0.7811	$1.1369e^{-13}$
$(200, 77)$	$(96, 67)$	0.4822	$7.1154e^{-14}$
	$(96, 74)$	0.5032	$5.6042e^{-14}$
	$(137, 67)$	0.5321	$1.5632e^{-13}$
	$(123, 72)$	0.5036	$7.1050e^{-14}$
$(200, 45)$	$(108, 45)$	0.2810	$1.4211e^{-14}$
	$(85, 40)$	0.2751	$3.5527e^{-14}$
	$(80, 45)$	0.2661	$7.1053e^{-14}$
	$(95, 38)$	0.2887	0
$(200, 20)$	$(72, 20)$	0.1338	0
	$(43, 18)$	0.1259	0
	$(56, 17)$	0.1294	$7.1054e^{-14}$
	$(80, 19)$	0.1135	$2.1316e^{-14}$

TABLE 2. Random graph with $N = 500$ fixed

(N, M)	(n, m)	CPU time	Error
$(500, 350)$	$(330, 283)$	16.6764	0
	$(414, 327)$	22.5961	$1.7053e^{-13}$
	$(451, 329)$	22.0645	$1.1369e^{-14}$
	$(447, 338)$	20.7785	$5.6843e^{-14}$
$(500, 265)$	$(461, 257)$	14.5689	$3.9790e^{-13}$
	$(476, 263)$	15.0924	$5.1159e^{-13}$
	$(468, 258)$	12.8741	$5.1158e^{-13}$
	$(460, 258)$	13.7202	$4.5475e^{-13}$
$(500, 155)$	$(315, 149)$	3.4211	$1.9875e^{-13}$
	$(298, 146)$	3.3781	$1.7153e^{-13}$
	$(349, 148)$	3.8173	$8.5265e^{-14}$
	$(297, 144)$	3.2799	$1.1369e^{-14}$
$(500, 100)$	$(123, 71)$	1.3621	$1.2412e^{-14}$
	$(107, 70)$	1.2230	$1.4211e^{-14}$
	$(122, 74)$	1.1793	$1,4211e^{-14}$
	$(136, 67)$	1.1785	$1.1369e^{-13}$
$(500, 70)$	$(140, 59)$	0.7183	$2.9433e^{-14}$
	$(139, 59)$	0.7111	$2.9433e^{-14}$
	$(160, 52)$	0.7243	$1.7053e^{-13}$
	$(121, 63)$	0.6915	$2.8421e^{-14}$
$(500, 20)$	$(114, 18)$	0.1380	$2.8511e^{-14}$
	$(120, 18)$	0,1510	$2.1316e^{-14}$
	$(94, 15)$	0.1418	$7.1055e^{-15}$
	$(128, 18)$	0.1625	$7.1055e^{-15}$

TABLE 3. Maximum bi-clique in random graphs

5.3. **Bi-cliques in real-world bipartite graphs.** Our experiments include a few real-world graphs from the KONECT project[1], which focuses on collecting network datasets. We have considered bipartite graphs for network datasets, for which the left nodes represent the users, and the right nodes represent the pages. Each edge indicates an edit. The results of the real graphs are provided in Table 4. With the value $\rho = 0.05$ in (10), we employ **Algorithm** 1 in the adjacency matrix of all bipartite graphs considered.

Here, (N, M) represents the number of vertices on the left and right, respectively. The corresponding recovered bi-clique sizes are shown in the column under (n, m). The last column includes the runtime in seconds.

For all the seven datasets considered in Table 4, all bi-cliques are found with errors occurring on $[10^{-15}, 10^{-13}]$ using our approach. For comparison purposes, we have also run DSA with input $n \times m$ from column 4, Table 4. The errors for all seven problems are in $[10^{-5}, 10^{-3}]$. Despite the fact that DSA requires $n \times m$ as input to the program, the errors produced are far from satisfactory.

Our results demonstrate that the proposed algorithm not only recovers planted bi-cliques with high accuracy but also scales effectively to real-world bipartite networks. This aligns with recent advances in large-scale bi-clique enumeration, such as the efficient counting techniques developed by Chen et al. [10], which highlight the feasibility of handling large bipartite structures in practice. Moreover, the convex programming perspective adopted by Zhou et al. [26] for anchored densest subgraph

[1] http://konect.cc/.

Graph	(N, M)	Number of edges	(n, m)	Runtime
Wikiquote edits (am)	$(138, 417)$	630	$(1, 168)$	3.0057
Wikibooks edits (co)	$(73, 256)$	500	$(1, 92)$	0.4479
Wikibooks edits (uz)	$(102, 398)$	565	$(1, 163)$	1.3830
Wikibooks edits (za)	$(48, 142)$	233	$(1, 46)$	0.4917
Wikipedia edits (atj)	$(83, 572)$	6306	$(4, 360)$	16.4703
Wikipedia edits (aa)	$(182, 507)$	1189	$(2, 97)$	4.0664
Wikiquote edits (ang)	$(109, 516)$	1303	$(2, 205)$	5.3436

TABLE 4. Maximum bi-cliques in real-world graphs

search further supports the validity of convex models for the recovery of dense substructures in noisy settings. Compared to these methods, our formulation offers a unified, decomposition-based approach that recovers binary structure from noisy data with minimal tuning.

6. Conclusion.

We have suggested a mathematical model for the bi-clique problem that differs from the regular matrix decomposition model. A benefit of our model is that it produces binary solutions for the entries of the solution matrices. This was achieved using the weighted ℓ_1-norm. Our algorithm does not require input from the user other than the adjacency matrix of the input graph. In addition, the algorithm can be easily implemented without needing any external solvers. Although the algorithm has been proposed for the planted edge bi-clique problem, it has been tested on the MEB problem using random graphs with almost error-free results. We also applied our algorithm to several real-world bipartite graphs, and bi-cliques were successfully recovered.

Limitations and future work. While the proposed model demonstrates strong empirical performance, several limitations remain. First, the theoretical advantage of the weighted ℓ_1-norm over the standard ℓ_1-norm is not formally established within a convex optimization framework; its benefits are currently supported by intuitive analysis and numerical evidence. Second, the computational cost may increase for very large graphs due to repeated singular value thresholding.

Future work will aim to provide a rigorous theoretical comparison of norm choices, enhance scalability, and apply the model to specific real-world problems such as recommender systems, gene-condition interactions, or fraud detection, where solving the MEB problem has significant practical value. In particular, future research should aim to formally explain why the weighted ℓ_1-norm more strongly promotes sparsity, potentially by analyzing its adaptive penalization mechanism and its impact on the optimization landscape compared to the standard ℓ_1-norm.

Acknowledgments. This work was supported by the Organization for Women in Science for the Developing World (OWSD) and the Swedish International Development Cooperation Agency (Sida).

REFERENCES

[1] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P. L. Hammer and B. Simeone, Consensus algorithms for the generation of all maximal bicliques, *Discrete Applied Mathematics*, **145** (2004), 11-21.

[2] C. Ambühl, M. Mastrolilli and O. Svensson, Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut, *SIAM Journal on Computing*, **40** (2011), 567-596.
[3] B. Ames, Convex relaxation for the planted clique, biclique, and clustering problems.
[4] B. P. W. Ames and S. A. Vavasis, Nuclear norm minimization for the planted clique and biclique problems, *Mathematical Programming*, **129** (2011), 69-89.
[5] D. Bertsimas, R. Cory-Wright and J. Pauphilet, Sparse plus low-rank matrix decomposition: A discrete optimization approach, arXiv preprint. arXiv:2109.12701.
[6] P. Bombina and B. Ames, Convex optimization for the densest subgraph and densest submatrix problems, *Operations Research Forum*, **1** (2020), Paper No. 23, 24 pp.
[7] J.-F. Cai, E. J. Candès and Z. Shen, A singular value thresholding algorithm for matrix completion, *SIAM Journal on Optimization*, **20** (2010), 1956-1982.
[8] E. J. Candès, X. Li, Y. Ma and J. Wright, Robust principal component analysis?, *Journal of the ACM (JACM)*, **58** (2011), Art. 11, 37 pp.
[9] E. J. Candes, M. B. Wakin and S. P. Boyd, Enhancing sparsity by reweighted l_1 minimization, *Journal of Fourier Analysis and Applications*, **14** (2008), 877-905.
[10] Y. Chen, X. Wang and J. Sun, Efficient biclique counting in large bipartite graphs, in *Proceedings of the ACM on Management of Data (SIGMOD)*, ACM, 2023, 2136-2149.
[11] V. M. F. Dias, C. M. H. de Figueiredo and J. L. Szwarcfiter, On the generation of bicliques of a graph, *Discrete Applied Mathematics*, **155** (2007), 1826-1832.
[12] N. Gillis and F. Glineur, Nonnegative factorization and the maximum edge biclique problem, arXiv preprint. arXiv:0810.4225.
[13] N. Gillis and F. Glineur, A continuous characterization of the maximum-edge biclique problem, *Journal of Global Optimization*, **58** (2014), 439-464.
[14] L. Han, S. Bi and S. Pan, Two-stage convex relaxation approach to least squares loss constrained low-rank plus sparsity optimization problems, *Computational Optimization and Applications*, **64** (2016), 119-148.
[15] D. S. Johnson, The NP-completeness column: an ongoing guide, *Journal of Algorithms*, **6** (1985), 434-451.
[16] T. Kloks and D. Kratsch, Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, *Information Processing Letters*, **55** (1995), 11-16.
[17] B. Lyu, L. Qin, X. Lin, Y. Zhang, Z. Qian and J. Zhou, Maximum biclique search at billion scale, *Proceedings of the VLDB Endowment*.
[18] P. Manurangsi, Inapproximability of maximum edge biclique, maximum balanced biclique and minimum k-cut from the small set expansion hypothesis, in *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017, Art. No. 79, 14 pp.
[19] D. Nussbaum, S. Pu, J.-R. Sack, T. Uno and H. Zarrabi-Zadeh, Finding maximum edge bicliques in convex bipartite graphs, *Algorithmica*, **64** (2012), 311-325.
[20] N. Parikh and S. Boyd, Proximal algorithms, *Foundations and Trends in Optimization*, **1** (2014), 127-239.
[21] R. Peeters, The maximum edge biclique problem is np-complete, *Discrete Applied Mathematics*, **131** (2003), 651-654.
[22] O. Salma, Harnessing the mathematics of matrix decomposition to solve the planted and maximum clique problems, School of Computer Science and Applied Mathematics, University of the Witwatersrand, South Africa, unpublished paper, 2022.
[23] M. Sözdinler and C. Özturan, Finding maximum edge biclique in bipartite networks by integer programming, in *2018 IEEE International Conference on Computational Science and Engineering (CSE)*, IEEE, (2018), 132-137.
[24] M. Yannakakis, Node-and edge-deletion np-complete problems, in *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, (1978), 253-264.
[25] P. Zhai and S. Zhang, Learnable graph-regularization for matrix decomposition, *ACM Transactions on Knowledge Discovery from Data (TKDD)*, **17** (2023), 32.
[26] R. Zhou, H. Zhang, X. Fang and J. Huang, Efficient and effective anchored densest subgraph search: A convex-programming based approach, in *Proceedings of the 30th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2024.

Received March 2025; revised August 2025; early access August 2025.