A Scalable and Exact Relaxation for Densest k-Subgraph via Error Bounds

Ya Liu[1*], **Junbin Liu** [1*], **Wing-Kin Ma**[1], **Aritra Konar**[2†]

[1] The Chinese University of Hong Kong
[2] KU Leuven
yaliu@link.cuhk.edu.hk, liujunbin@link.cuhk.edu.hk, wkma@ee.cuhk.edu.hk, aritra.konar@kuleuven.be

Abstract

Given an undirected graph and a size parameter k, the Densest k-Subgraph (DkS) problem extracts the subgraph on k vertices with the largest number of induced edges. While DkS is NP–hard and difficult to approximate, penalty-based continuous relaxations of the problem have recently enjoyed practical success for real-world instances of DkS. In this work, we propose a scalable and exact continuous penalization approach for DkS using the error bound principle, which enables the design of suitable penalty functions. Notably, we develop new theoretical guarantees ensuring that both the global and local optima of the penalized problem match those of the original problem. The proposed penalized reformulation enables the use of first-order continuous optimization methods. In particular, we develop a non-convex proximal gradient algorithm, where the non-convex proximal operator can be computed in closed form, resulting in low per-iteration complexity. We also provide convergence analysis of the algorithm. Experiments on large-scale instances of the DkS problem and one of its variants, the Densest (k_1, k_2) Bipartite Subgraph (Dk_1k_2BS) problem, demonstrate that our method achieves a favorable balance between computation cost and solution quality.

Code — https://github.com/ly6421/dks-aaai2026
Extended version — https://arxiv.org/abs/2511.11451

Introduction

Dense subgraph discovery (DSD) is a key graph mining primitive which aims to extract highly interconnected subsets of vertices from a given graph (see (Lanciano et al. 2024; Luo et al. 2023) and the references therein). The problem has garnered considerable attention as it finds myriad applications ranging from mining trending topics in social media (Angel et al. 2014), discovering complex patterns in gene annotation graphs (Saha et al. 2010), to detecting fraudsters in e-commerce and financial transaction networks (Hooi et al. 2016; Li et al. 2020; Chen and Tsourakakis 2022).

The broad applicability of DSD has resulted in the development of several formulations and algorithms. Notable among these is the Densest Subgraph (DSG) problem (Goldberg 1984), which extracts the subgraph with the maximum average induced degree. Although DSG can be solved exactly in polynomial time via maximum flow, it is computationally expensive. A faster greedy peeling algorithm runs in linear time and provides a 0.5-factor approximation guarantee (Charikar 2000). Improved multi-stage extensions of the greedy algorithm were later developed in (Boob et al. 2020; Chekuri, Quanrud, and Torres 2022) - these offer improved approximation guarantees for DSG at the same complexity order. A separate line of research (Danisch, Chan, and Sozio 2017; Harb, Quanrud, and Chekuri 2022, 2023; Nguyen and Ene 2024) has developed algorithms for DSG using tools from convex optimization, building on a linear programming relaxation of DSG due to (Charikar 2000). A different type of dense subgraph is the max-core of a graph (Seidman 1983), which corresponds to the subgraph that maximizes the minimum induced degree. The greedy algorithm for DSG optimally solves this problem in linear time. DSG and the max-core were later unified within a general framework for computing solutions of degree-based density objectives in (Veldt, Benson, and Kleinberg 2021).

A limitation of the DSG and the max-core formulations is that they do not allow the size of the extracted subgraph to be explicitly controlled, which can result in the output being a large subgraph with poor inter-connectivity in real-world graphs, as observed in (Tsourakakis et al. 2013; Shin, Eliassi-Rad, and Faloutsos 2016). Indeed, by design, the solution of DSG is maximal in size (Harb, Quanrud, and Chekuri 2023). Furthermore, as the solution of both problems is unique, there is no flexibility in extracting a denser subgraph of a different size. Arguably, the simplest remedy is to add an explicit size constraint to DSG, which results in the Densest k-Subgraph (DkS) problem (Feige, Peleg, and Kortsarz 2001). Variation of the size parameter k enables extraction of dense subgraphs of different sizes, thereby allowing users to select a solution with the desired density. The flip side of this flexibility is that DkS is NP-hard and difficult to approximate in the worst case (Manurangsi 2017). However, practical optimization algorithms have been demonstrated to exhibit good performance on real-world instances of the problem (Papailiopoulos et al. 2014; Konar and Sidiropoulos 2021; Lu, Sidiropoulos, and Konar 2025; Liu et al. 2024b). Our present work advances

[*] These authors contributed equally.
[†] Corresponding author.

this line of research by developing a new principled and scalable continuous optimization framework for effectively tackling such "favorable" instances of DkS.

Variations of DkS which allow extraction of subgraphs of different sizes also exist - these include the Densest at-least-k Subgraph (Dal$k$S) and Densest at-most-k Subgraph (Dam$k$S) problems (Andersen and Chellapilla 2009; Khuller and Saha 2009), and the f-densest subgraph problem (Kawase and Miyauchi 2018). However, in contrast to DkS, these formulations are not guaranteed to span the spectrum of densest subgraphs of *every* size.

Contributions: Our main contributions are summarized as follows:

• We propose a scalable and exact penalization formulation for the DkS problem via the error bound principle (Luo, Pang, and Ralph 1996; Cui and Pang 2021). By designing a suitable relaxation of the constraint set of DkS and a corresponding error bound function that serves as a penalty, we arrive at a non-convex, non-smooth penalty formulation. In the context of penalty methods for constrained optimization, the challenge lies in showing whether a penalty formulation at hand has its globally optimal solution set identical to that of the original problem. In this work, we established not only the equivalence of global optima between our formulations and DkS, but also that of local optima.

• To tackle our proposed formulation, we employ a non-convex variant of the classic proximal gradient algorithm (Nesterov 2018), comprising a "forward" gradient step, and a "backward" proximal operator evaluation. A key technical challenge is that evaluating the proximal operator in our formulation entails solving a non-convex problem. Nevertheless, it surprisingly turns out that the global solution of this subproblem can be computed efficiently even in the absence of convexity, resulting in low per-iteration complexity, and thereby endowing the algorithm with scalability. Theoretically, we also show that the proposed algorithm is guaranteed to converge to a critical point of our formulation at a sub-linear rate.

• We evaluate our method on the DkS problem and adapt our method to the Densest (k_1, k_2) Bipartite Subgraph (Dk_1k_2BS) problem as well. Extensive experiments on large-scale real-world graphs—with sizes ranging from thousands to millions of nodes—show that our method consistently achieves state-of-the-art performance with low computational cost.

Notation: Let $\mathbb{R}$ denote the set of real numbers. Vectors and matrices are represented by bold lowercase (e.g., $\boldsymbol{x}$) and bold uppercase letters (e.g., $\boldsymbol{X}$), respectively. The ith element of $\boldsymbol{x}$ is x_i, and the (i,j)th entry of $\boldsymbol{X}$ is x_{ij}. The transpose of $\boldsymbol{x}$ is denoted by $\boldsymbol{x}^\top$. We use $\mathbf{0}$ and $\mathbf{1}$ to denote the all-zero and all-one vectors, and $\boldsymbol{I}$ for the identity matrix. The ith largest entry of $\boldsymbol{x} \in \mathbb{R}^n$ is denoted by $x_{[i]}$, and the *max-k-sum* is defined as $S_k(\boldsymbol{x}) := x_{[1]} + \cdots + x_{[k]}$. The notation $\Pi_{\mathcal{X}}(\boldsymbol{x})$ denotes the projection of $\boldsymbol{x}$ onto the set $\mathcal{X}$.

Related Work

Owing to its intrinsic difficulty, DkS has been tackled from different perspectives, aiming to strike a good balance between computational cost and solution quality. The state-of-the-art polynomial-time approximation algorithm for DkS due to (Bhaskara et al. 2010) outputs an $O(n^{1/4+\epsilon})$-approximation solution in time $O(n^{1/\epsilon})$ for every $\epsilon > 0$, which is quite pessimistic. Alternative schemes include hierarchies of linear programming relaxations (Bhaskara et al. 2012) and semidefinite programming (SDP) relaxations (Feige, Seltser et al. 1997; Karisch, Rendl, and Clausen 2000; Bombina and Ames 2020). While these approaches offer certain optimality guarantees, they are computationally expensive for large-scale problems, mainly due to the fact that these relaxations employ extra variables, thereby increasing complexity.

More practical approaches for DkS include greedy algorithms (Feige, Peleg, and Kortsarz 2001; Asahiro et al. 2000); they are computationally efficient but often yield suboptimal solutions due to their inherently myopic nature. Meanwhile, the Truncated Power Method (TPM) of (Yuan and Zhang 2013) is a scalable method that converges under certain conditions. However, it was found in practice that the subgraphs obtained by TPM can be highly sub-optimal. Another approach is the Spannogram of (Papailiopoulos et al. 2014), which uses a low-rank factorization of the graph adjacency matrix to approximately solve the DkS problem. Using higher rank approximation improves the solution quality, but at the expense of a larger running time, owing to the exponential dependence of the algorithm's complexity on the rank. For example, using just a rank-2 approximation incurs $O(n^3)$ complexity, which limits scalability.

A separate line of research has blossomed around developing different continuous relaxations of DkS followed by application of continuous optimization algorithms - these range from non-convex (Hager, Phan, and Zhu 2016; Sotirov 2020) to convex formulations (Konar and Sidiropoulos 2021). However, none of these works analyzed whether their respective relaxations are tight or not. Closest to our present work are the recent exact penalty-based methods of (Liu, Liu, and Ma 2024; Lu, Sidiropoulos, and Konar 2025; Liu et al. 2024a,b), which have demonstrated promising performance in addressing large-scale DkS problems. The general principle behind these approaches is to relax the combinatorial constraint set of DkS to arrive at a continuous optimization problem and employ penalty terms to enforce the desired combinatorial constraints. Through different lines of reasoning, these works establish equivalence between the globally optimal solutions of their formulations and those of DkS. First-order continuous optimization algorithms are adopted for the resulting formulations, leading to low computational cost. Although our approach falls in the above category of methods, it offers several benefits over the prevailing state-of-the-art (see Contributions and **(D1)-(D2)**) in the next section.

Background and Preliminaries

Let $\mathcal{G} = (\mathcal{V}_n, \mathcal{E}_m)$ be an undirected, unweighted graph with $\mathcal{V}_n$ being the vertex set containing n vertices and $\mathcal{E}_m$ being the edge set containing m edges. Given a size parameter $k < n$, the DkS problem aims to identify a vertex set

$\mathcal{S}_k \subseteq \mathcal{V}_n$ with k vertices such that the number of edges between vertices in $\mathcal{S}_k$ is maximized. Formally, the problem can be expressed as

$$\max_{\boldsymbol{x}\in\mathbb{R}^n} \quad \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}, \quad \text{s.t. } \boldsymbol{x} \in \mathcal{U}_k^n \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ denotes the adjacency matrix of $\mathcal{G}$, and $\mathcal{U}_k^n := \{\boldsymbol{x} \in \{0,1\}^n \mid \boldsymbol{x}^\top \mathbf{1} = k\}$ is a set of binary vectors with k nonzero entries. We term the set $\mathcal{U}_k^n$ as the *selection vector set*. In this work, we adopt a continuous optimization approach for the purpose of obtaining good approximate solutions to (1). The theoretical underpinnings are based on the concept of *exact penalization*, which is formally defined below.

Definition 1 (Exact Penalization) *Consider the following constrained optimization problem:*

$$\min_{\boldsymbol{x}\in\mathcal{A}} f(\boldsymbol{x}), \tag{2}$$

where $f : \mathcal{D} \to \mathbb{R}$ is the objective function defined on a domain $\mathcal{D} \subseteq \mathbb{R}^n$, and $\mathcal{A} \subseteq \mathcal{D}$ is a non-empty constraint set. Assume that the optimal solution set of (2) *is non-empty. Now consider the penalization formulation:*

$$\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}) + \lambda h(\boldsymbol{x}), \tag{3}$$

where $\mathcal{X} \subseteq \mathcal{D}$ is a constraint set such that $\mathcal{A} \subseteq \mathcal{X}$, $h : \mathbb{R}^n \to \mathbb{R}$ is a penalty function for enforcing the constraint satisfaction of $\mathcal{A}$, and $\lambda > 0$ is a given penalty parameter. If there exists a finite λ such that the optimal solution set of (3) *is the same as that of* (2)*, then* (3) *is called an exact penalization of* (2)*.*

An instance of exact penalization for the DkS problem is the following formulation

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} -\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} - \lambda\|\boldsymbol{x}\|_2^2, \text{ s.t. } \boldsymbol{x} \in \text{conv}\left(\mathcal{U}_k^n\right), \tag{4}$$

where $\text{conv}\left(\mathcal{U}_k^n\right) = \{\boldsymbol{x} \in [0,1]^n \mid \boldsymbol{x}^\top \mathbf{1} = k\}$ is the convex hull of $\mathcal{U}_k^n$ and $\lambda > 0$ is the penalization parameter. The *exact penalization* property of (4) was shown from different perspectives. The study in (Liu et al. 2024a,b) showed exactness through both the concave minimization principle and the error bound principle (Luo, Pang, and Ralph 1996; Cui and Pang 2021). The study in (Lu, Sidiropoulos, and Konar 2025) leveraged an extension of the Motzkin-Straus theorem (Motzkin and Straus 1965) to prove exactness. Formulation (4) is a non-convex optimization problem with a convex constraint, and hence, it can be readily tackled by first-order optimization algorithms. The work (Lu, Sidiropoulos, and Konar 2025) employed the Frank-Wolfe algorithm, and the paper (Liu et al. 2024b) adopted projected gradient descent combined with majorization-minimization. However, prior studies working with problem (4) have their drawbacks.

(D1): As the optimization landscape of (4) is non-convex, optimization algorithms may converge to local minima, which are not guaranteed to be feasible for DkS; i.e., they may fall outside the constraint set $\mathcal{U}_k^n$. Therefore, establishing equivalence between the sets of local optima of the penalized and original formulations is of practical value. While the work (Lu, Sidiropoulos, and Konar 2025) proves global optimality equivalence, the relationship between local optima was not addressed. The result in (Liu et al. 2024a,b) essentially suggests that formulation (4) has both global and local solution equivalence. The limitation of this prior result is however that we need the convex hull $\text{conv}(\mathcal{U}_n^k)$ as the constraint set, which means that the result may no longer apply if we consider a different relaxation set.

(D2): As mentioned, the existing studies consider the convex hull of $\mathcal{U}_k^n$ as the relaxation set. We are concerned with the computational aspects arising from the use of $\text{conv}(\mathcal{U}_n^k)$ as the constraint set. The projected gradient methods require projecting onto $\text{conv}(\mathcal{U}_k^n)$, which does not have a closed-form solution and requires a bisection search with complexity $\mathcal{O}(n \log n)$ (Konar and Sidiropoulos 2021). As the problem size increases, the associated computational cost can become non-negligible. The Frank-Wolfe Algorithm (Lu, Sidiropoulos, and Konar 2025) has a per iteration cost of $\mathcal{O}(n \log k)$. In practice, however, it was noted that the Frank-Wolfe Algorithm can be slow compared to the proximal gradient methods (which will be reflected in our numerical results). This motivates the exploration of alternative, easier-to-work-with relaxation sets with appropriate penalization functions that maintain exactness for both global and local optimal solution sets and support efficient algorithm design.

The Error Bound Principle: The error bound principle (Luo, Pang, and Ralph 1996; Cui and Pang 2021) is a general optimization technique that can be leveraged for exact penalization. The same principle was also used to develop the exact penalization (4) for DkS in (Liu et al. 2024a,b). However, as we will see later, our invocation of error bounds results in a penalization formulation that differs significantly from that of (Liu et al. 2024a,b).

To introduce the error bound principle, we need the notion of error bound functions.

Definition 2 (Error Bound Function) *Given a set $\mathcal{X} \subseteq \mathbb{R}^n$ and a set $\mathcal{A} \subseteq \mathcal{X}$, a function $\psi : \mathbb{R}^n \to \mathbb{R}$ is said to be an error bound function of $\mathcal{A}$ relative to $\mathcal{X}$ if*

$$\text{dist}(\boldsymbol{x}, \mathcal{A}) \leq \psi(\boldsymbol{x}), \qquad \forall \boldsymbol{x} \in \mathcal{X}, \tag{5}$$

$$\psi(\boldsymbol{x}) = 0, \qquad \forall \boldsymbol{x} \in \mathcal{A}, \tag{6}$$

where dist *is defined as*

$$\text{dist}(\boldsymbol{x}, \mathcal{A}) := \inf_{\boldsymbol{y}\in\mathcal{A}} \|\boldsymbol{x} - \boldsymbol{y}\|_2.$$

An error bound function provides an upper bound on the distance from a point to the target set $\mathcal{A}$. Intuitively, when used as a penalty function, it promotes feasibility by encouraging solutions to lie within the target set $\mathcal{A}$. The following lemma formalizes this intuition, showing that error bound functions can yield exact penalization.

Lemma 1 (Theorem 2.1.2 in (Luo, Pang, and Ralph 1996), or Proposition 9.1.1 in (Cui and Pang 2021)) *Let $\mathcal{D} \subseteq \mathbb{R}^n$. Let $\mathcal{A} \subseteq \mathcal{D}$ be a non-empty closed set. Let $f : \mathcal{D} \to \mathbb{R}$ be a function that is K-Lipschitz continuous on some set $\mathcal{X} \subseteq \mathcal{D}$, with $\mathcal{A} \subseteq \mathcal{X}$. Suppose that*

$$\min_{\boldsymbol{x}\in\mathcal{A}} f(\boldsymbol{x}) \tag{7}$$

has an optimal solution. Let ψ be an error bound function of $\mathcal{A}$ relative to $\mathcal{X}$. Given any scalar $\lambda > K$, the following problem

$$\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}) + \lambda\,\psi(\boldsymbol{x}) \tag{8}$$

is an exact penalization of problem (7).

This lemma establishes that for problems with Lipschitz continuous objective functions, exact penalization can be achieved by constructing a valid error bound function ψ and selecting a sufficiently large but finite penalty parameter λ. Notably, many functions—including the objective in the DkS formulation (1)—are Lipschitz continuous over compact sets. An important aspect of this approach is that the error bound function is intrinsically tied to the choice of the relaxation set $\mathcal{X}$. Unlike traditional penalty methods such as the proximal distance method (Keys, Zhou, and Lange 2019) and the quadratic penalty method (Nocedal and Wright 2006), which reformulate the original problem as an unconstrained minimization with penalty terms, the error bound approach emphasizes the design of both the relaxation set and the penalty function. This often results in more structured and tractable penalization formulations. As mentioned previously, the method in (Liu et al. 2024a,b) is one such example. It chooses conv($\mathcal{U}_k^n$) as the relaxation set. The objective function $f(\boldsymbol{x}) = -\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}$ is $2\sqrt{k}\|\boldsymbol{A}\|_2$-Lipschitz continuous over the set conv($\mathcal{U}_k^n$). It can then be shown that $\psi(\boldsymbol{x}) = 2(k - \|\boldsymbol{x}\|_2^2)$ is an error bound function of $\mathcal{U}_k^n$ relative to its convex hull conv($\mathcal{U}_k^n$). This leads to the formulation (4).

Proposed Approach

We develop a new penalization approach for tackling DkS by leveraging the error bound principle. For this task, we need two ingredients: (i) a relaxation set of the constraints $\mathcal{U}_k^n$, and (ii) a suitable error bound function. For the first part, instead of relaxing $\mathcal{U}_k^n$ to its convex hull conv($\mathcal{U}_k^n$), we relax it to the hypercube $[0,1]^n$, which admits a simpler description. The next step is to construct an algorithmically tractable error bound function, which serves as a penalty to encourage points in $[0,1]^n$ to move toward the selection vector set $\mathcal{U}_k^n$. To this end, we have the following result.

Lemma 2 *For any $\boldsymbol{x} \in [0,1]^n$, the following inequality holds:*

$$\operatorname{dist}(\boldsymbol{x}, \mathcal{U}_k^n) \le \psi(\boldsymbol{x}) := k + \mathbf{1}^\top \boldsymbol{x} - 2S_k(\boldsymbol{x}), \tag{9}$$

where $S_k(\boldsymbol{x}) = \sum_{i=1}^k x_{[i]}$, and $x_{[1]} \ge x_{[2]} \ge \cdots \ge x_{[n]}$ denotes the sorted components of $\boldsymbol{x}$ in descending order. Also, $\psi(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \mathcal{U}_k^n$. The function $\psi(\boldsymbol{x})$ is an error bound function of $\mathcal{U}_k^n$ relative to $[0,1]^n$.

Proof of Lemma 2: Let $\boldsymbol{x} \in [0,1]^n$. Without loss of generality, assume $x_1 \ge x_2 \ge \cdots \ge x_n$. Let $\boldsymbol{y} = \Pi_{\mathcal{U}_k^n}(\boldsymbol{x})$ denote the projection of $\boldsymbol{x}$ onto $\mathcal{U}_k^n$. Then, $\boldsymbol{y} = (1,\ldots,1,0,\ldots,0)$ with the first k elements being 1. The distance function can be upper bounded as:

$$\begin{aligned}
\operatorname{dist}(\boldsymbol{x}, \mathcal{U}_k^n) &= \|\boldsymbol{x} - \boldsymbol{y}\|_2 \le \|\boldsymbol{x} - \boldsymbol{y}\|_1 \\
&= \sum_{i=1}^k (1 - x_i) + \sum_{i=k+1}^n x_i \\
&= k + \sum_{i=1}^n x_i - 2\sum_{i=1}^k x_i \\
&= k + \mathbf{1}^\top \boldsymbol{x} - 2S_k(\boldsymbol{x}).
\end{aligned} \tag{10}$$

When $\boldsymbol{x} \in \mathcal{U}_k^n$, we have

$$\operatorname{dist}(\boldsymbol{x}, \mathcal{U}_k^n) = k + \mathbf{1}^\top \boldsymbol{x} - 2S_k(\boldsymbol{x}) = k + k - 2k = 0.$$

The proof is complete. ■

Based on the error bound function $\psi(\boldsymbol{x})$ in (9), we propose the following penalized reformulation of the DkS problem (1):

$$\min_{\boldsymbol{x}\in[0,1]^n} F_\lambda(\boldsymbol{x}) := \underbrace{-\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}}_{=f(\boldsymbol{x})} + \lambda \underbrace{\left[\mathbf{1}^\top \boldsymbol{x} - 2S_k(\boldsymbol{x})\right]}_{:=h(\boldsymbol{x})}. \tag{11}$$

It is straightforward to verify that the objective function $f(\cdot)$ is $2\sqrt{n}\|\boldsymbol{A}\|_2$-Lipschitz continuous over $[0,1]^n$. Consequently, exact penalization holds by Lemma 1. More importantly, formulation (11) also has the local optimal solution set equivalence, as stated in the following proposition.

Proposition 1 *Consider the DkS problem (1) and its penalized reformulation* (11)*. Given any scalar $\lambda > 2\sqrt{n}\|\boldsymbol{A}\|_2$, problem* (11) *is an equivalent formulation of problem* (1) *in the sense that both the* **global** *and* **local** *optimal solution sets of* (11) *are the same as the corresponding sets of* (1)*.*

The proof of Proposition 1 is relegated to Part A of the supplementary material. Compared to Lemma 1, Proposition 1 extends the result from the equivalence of global optimal solution sets to the equivalence of local optimal solution sets *without any* additional assumptions. This offers a practical guarantee: if an algorithm designed to solve (11) converges to a local minimizer, then this solution lies in $\mathcal{U}_k^n$ and is also a local minimizer of the original problem (1). Moreover, Proposition 1 is not limited to the DkS problem; it is applicable to a broader class of optimization problems constrained by the selection vector set, provided that the objective function is Lipschitz continuous on $[0,1]^n$.

A Non-Convex Proximal Gradient Algorithm: In this subsection, we develop a non-convex, first-order algorithm for solving (11). Since the cost function of (11) has a composite form comprising both differentiable and non-differentiable terms, we consider the proximal gradient method (PGM), whose iterations are given by (Nesterov 2018):

$$\begin{aligned}
\boldsymbol{z}^\ell &= \boldsymbol{x}^\ell + \gamma_\ell(\boldsymbol{x}^\ell - \boldsymbol{x}^{\ell-1}) \\
\boldsymbol{x}^{\ell+1} &= \operatorname{prox}_{[0,1]^n, \eta_\ell\lambda h}(\boldsymbol{z}^\ell - \eta_\ell \nabla f(\boldsymbol{z}^\ell)),
\end{aligned} \tag{12}$$

where $l = 0, 1, ...,$ index iterations, $\boldsymbol{x}^0 = \boldsymbol{x}^{-1}$, $\{\eta_\ell\}_{\ell\ge 0}$ is a step size sequence, $\{\gamma_\ell\}_{\ell\ge 0}$ is an extrapolation sequence and we adopt the FISTA extrapolation (Beck and Teboulle

2009); $\text{prox}_{[0,1]^n,\eta_\ell\lambda h}$ denotes the proximal operator and is defined as

$$\text{prox}_{[0,1]^n,\eta_\ell\lambda h}(\boldsymbol{z}) = \underset{\boldsymbol{x}\in[0,1]^n}{\arg\min}\ \frac{1}{2}\|\boldsymbol{z}-\boldsymbol{x}\|_2^2+\eta_\ell\lambda h(\boldsymbol{x}). \quad (13)$$

Note that we add extrapolation in the PGM, which in practice was found to lead to faster convergence. Computing the gradient of $f(\cdot)$ in each iteration entails performing a sparse-matrix vector multiplication, which can be accomplished in time $O(m)$ where m is the number of edges. A critical consideration for the PGM is whether the proximal operator can be computed efficiently. In our case, with $h(\boldsymbol{x}) = \mathbf{1}^\top\boldsymbol{x} - 2S_k(\boldsymbol{x})$, problem (13) is not guaranteed to be convex for every λ. Nevertheless, we show that despite the absence of convexity, a global optimum can still be computed in an efficient manner.

Proposition 2 *Consider the following non-convex optimization problem*

$$\min_{\boldsymbol{x}\in[0,1]^n}\ \tfrac{1}{2}\|\boldsymbol{z}-\boldsymbol{x}\|_2^2+\mu(\mathbf{1}^\top\boldsymbol{x}-2S_k(\boldsymbol{x})), \quad (14)$$

where $\mu > 0$ is given. Without loss of generality, assume $z_1 \geq z_2 \geq \cdots \geq z_n$. The following vector

$$\boldsymbol{x}^* = \left([z_1+\mu]_0^1,\ldots,[z_k+\mu]_0^1,[z_{k+1}-\mu]_0^1,\ldots,[z_n-\mu]_0^1\right)$$

is a global optimal solution to problem (14) where $[a]_0^1 := \min(1,\max(0,a))$ clips the variable to lie within $[0,1]$.

The proof is provided in Part B of the supplementary material. To evaluate the proximal operator, it suffices to identify the top-k entries of $\boldsymbol{z}$, which can be done in $\mathcal{O}(n\log k)$ time using a min-heap. This is generally faster than the $\mathcal{O}(n\log n)$ cost required to project onto $\text{conv}(\mathcal{U}_k^n)$ (Liu et al. 2024b; Konar and Sidiropoulos 2021), thereby enhancing scalability, particularly when identifying small subgraphs within extremely large graphs. This advantage will be demonstrated in the section on numerical results.

In addition to its low per-iteration computational cost, the proposed algorithm is guaranteed to converge to a critical point of Problem (11) at a sublinear rate. Here, a point $\boldsymbol{x}$ is defined as a critical point if

$$\mathbf{0} \in \partial F_\lambda(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) + \lambda\,\partial h(\boldsymbol{x}), \quad (15)$$

where $\partial h(\boldsymbol{x})$ denotes the limiting subdifferential of h at $\boldsymbol{x}$ (Rockafellar and Wets 2009; Li, So, and Ma 2020). The following convergence result is established.

Proposition 3 *Consider the PGM (12) for Problem (11). Suppose the step sizes satisfy $c_1 L_f \leq 1/\eta_\ell \leq c_2 L_f$ with $1 < c_1 \leq c_2 < \infty$, and the extrapolation weights satisfy $0 \leq \gamma_\ell < \bar{\gamma} < 1$ with $\bar{\gamma} = (c_1-1)/(2+2c_2)$. Then, the sequence $\{\boldsymbol{x}^\ell\}_{\ell\geq 0}$ generated by the PGM exhibits a sublinear convergence rate property*

$$\min_{\ell=0,\ldots,J}\ \text{dist}(\mathbf{0},\partial F_\lambda(\boldsymbol{x}^{\ell+1})) \leq \sqrt{C/(J+1)}, \quad (16)$$

where

$$C = \frac{64(1+c_2^2)(c_1-1)(F_\lambda(\boldsymbol{x}^0)-F_\lambda^*)\|\boldsymbol{A}\|_2}{(c_1-1)^2-4\bar{\gamma}^2(c_2+1)^2}$$

and $F_\lambda^* = \min_{\boldsymbol{x}\in[0,1]^n} F_\lambda(\boldsymbol{x})$.

Algorithm 1: The Proposed Algorithm for Solving (11)

Input: Initial point $\boldsymbol{x}^0$, initial penalty λ_0, an extrapolation sequence $\{\gamma_\ell\}_{\ell\geq 0}$, a step size sequence $\{\eta_\ell\}_{\ell\geq 0}$
Output: Final solution $\boldsymbol{x}^*$

$\boldsymbol{x}^{-1} = \boldsymbol{x}^0$
for $\ell = 0, 1, \ldots$ **do**
  $\boldsymbol{z}^\ell \leftarrow \boldsymbol{x}^\ell + \gamma_\ell(\boldsymbol{x}^\ell - \boldsymbol{x}^{\ell-1})$
  $\boldsymbol{x}^{\ell+1} \leftarrow \text{prox}_{[0,1]^n,\,\eta_\ell\lambda_\ell h}(\boldsymbol{z}^\ell - \eta_\ell\nabla f(\boldsymbol{z}^\ell))$
  if a penalty update condition is met **then**
    increase the penalty parameter: $\lambda_{\ell+1} > \lambda_\ell$
  end if
  if a stopping criterion is met **then**
    break
  end if
end for
return $\boldsymbol{x}^* = \boldsymbol{x}^{\ell+1}$

We provide the proof in Part C of the supplementary material. It should be noted that convergence analysis of the PGM with extrapolation and for a non-convex objective function was previously considered, e.g., in (Xu and Yin 2017). There, the convergence rate result requires the Kurdyka–Łojasiewicz (KL) property with the objective function f. The present convergence result differs in that we do not use the KL property, and we only use the Lipschitz continuous gradient property. We point out that the proposed non-convex PGM and its convergence analysis are applicable to other problems with the selection vector constraint, provided that the cost function has Lipschitz continuous gradients, and hence, it may prove to be of broader interest.

In practice, it has been observed that the algorithms may get stuck at poor local minima if the penalty parameter λ is set too large from the outset (Shao and Ma 2020; Zaslavskiy, Bach, and Vert 2008). To address this, λ is typically annealed from a small initial value to a larger one, eventually reaching the regime of exact penalization. We adopt this strategy in our implementation as well. The entire algorithm is summarized in Algorithm 1.

Adaptation to the Dk_1k_2BS Problem: The proposed approach can be easily adapted to the Dk_1k_2BS problem, a variant of the DkS problem. Given an undirected bipartite graph $\mathcal{G} = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}_m)$, where $\mathcal{V}_1$ and $\mathcal{V}_2$ are disjoint vertex sets with sizes $n_1 = |\mathcal{V}_1|$ and $n_2 = |\mathcal{V}_2|$, respectively, such that $n = n_1 + n_2$, the edge set $\mathcal{E}_m$ contains connections exclusively between nodes in $\mathcal{V}_1$ and $\mathcal{V}_2$. The goal of the Dk_1k_2BS problem is to select $k_1 < n_1$ vertices from $\mathcal{V}_1$ and $k_2 < n_2$ vertices from $\mathcal{V}_2$ such that the number of edges between the selected vertices—i.e., edges connecting one vertex in $\mathcal{V}_1$ to one in $\mathcal{V}_2$—is maximized. This problem also finds wide applications. For example, in recommendation systems, interactions between users and items can be naturally modeled as a bipartite graph, where identifying dense subgraphs helps reveal strong user-item affinities.

Formally, the Dk_1k_2BS problem can be formulated as:

$$\min_{\boldsymbol{x}\in\mathbb{R}^{n_1},\ \boldsymbol{y}\in\mathbb{R}^{n_2}} -\boldsymbol{x}^\top\boldsymbol{B}\boldsymbol{y}, \quad \text{s.t. } \boldsymbol{x}\in\mathcal{U}_{k_1}^{n_1},\ \boldsymbol{y}\in\mathcal{U}_{k_2}^{n_2}, \quad (17)$$

where $\boldsymbol{B}\in\mathbb{R}^{n_1\times n_2}$ denotes the biadjacency matrix for a bi-

Datasets	n	m
HepTh	9,877	25,998
CondMat	23,133	93,497
DBLP	317K	1M
roadNet	1.9M	2.7M
Talk	2.4M	5M
LiveJournal1	4.8M	69M

Table 1: Statistics of the datasets, where n represents the number of nodes and m represents the number of edges.

partite graph, and $\boldsymbol{x} \in \{0,1\}^{n_1}$, $\boldsymbol{y} \in \{0,1\}^{n_2}$ are selection vectors. Based on our error bound result, we can reformulate the Dk_1k_2BS problem as

$$\begin{aligned} \min_{\boldsymbol{x},\,\boldsymbol{y}} \quad & -\boldsymbol{x}^\top \boldsymbol{B}\boldsymbol{y} + \lambda\big(\mathbf{1}^\top \boldsymbol{x} + \mathbf{1}^\top \boldsymbol{y} - 2S_{k_1}(\boldsymbol{x}) - 2S_{k_2}(\boldsymbol{y})\big), \\ \text{s.t.} \quad & \boldsymbol{x} \in [0,1]^{n_1}, \quad \boldsymbol{y} \in [0,1]^{n_2}. \end{aligned} \tag{18}$$

The equivalence between (18) and (17), in the sense of Proposition 1, holds. This is because the error bounds for two disjoint sets can be combined to yield an error bound for their Cartesian product (Liu et al. 2024a). For implementation, we use the proposed proximal gradient method where the proximal operator is essentially the same as before. More details are provided in Part E of the supplementary material.

Numerical Results

In this section, we present empirical evaluations of our proposed algorithm on the DkS problem and the Dk_1k_2BS problem. We term our method, the exact penalization formulation solved by the proximal gradient method, as EP-Prox.

The DkS Problem

Implementation: We implement our method using Algorithm 1. The initial penalty parameter is set to $\lambda_0 = 10^{-10}$. The algorithm terminates when either $\|\boldsymbol{x}^{\ell+1} - \boldsymbol{x}^{\ell}\|_2^2 \leq 10^{-11}$, or the number of iterations exceeds 100. The penalty parameter λ is updated by $\lambda_{\ell+1} = 20\lambda_\ell$ when either $\|\boldsymbol{x}^{\ell+1} - \boldsymbol{x}^{\ell}\|_2 / \|\boldsymbol{x}^{\ell+1}\|_2 < 0.5$ or 10 iterations have passed since the last update.

Baselines: We used six widely-used baselines for DkS: (i) the greedy method (Feige, Peleg, and Kortsarz 2001); (ii) the truncated power method (TPM) (Yuan and Zhang 2013); (iii) rank-1 binary principal component (Rank-1 PC) approximation (Papailiopoulos et al. 2014); (iv) the Lovász relaxation via Linearized-ADMM (Lovász) (Konar and Sidiropoulos 2021); (v) the extreme point pursuit (EXPP) method (Liu et al. 2024a,b); (vi) the Frank-Wolfe (FW) method (Lu, Sidiropoulos, and Konar 2025). Information on the implementations of benchmark methods can be found in our GitHub repository. All the methods are initialized with the scaled all-one vector, $\boldsymbol{x}^0 = \mathbf{1}/n$.

Datasets: We evaluate the methods on a diverse set of real-world graphs obtained from (Leskovec and Krevl 2014). The detailed statistics of the datasets are summarized in Table 1. Each dataset was preprocessed by symmetrizing directed arcs (if the original dataset is a directed graph), removing all self loops, and extracting the largest connected component. To assess solution quality, we use the standard edge density metric:

$$\text{edge density} = \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}/(k(k-1)), \tag{19}$$

where $\boldsymbol{x} \in \mathcal{U}_k^n$ is the binary indicator vector representing the selected subgraph. Note that k is a fixed constant, and the numerator corresponds to the original objective function that we want to maximize in problem (1). Consequently, a larger edge density value reflects a higher-quality solution.

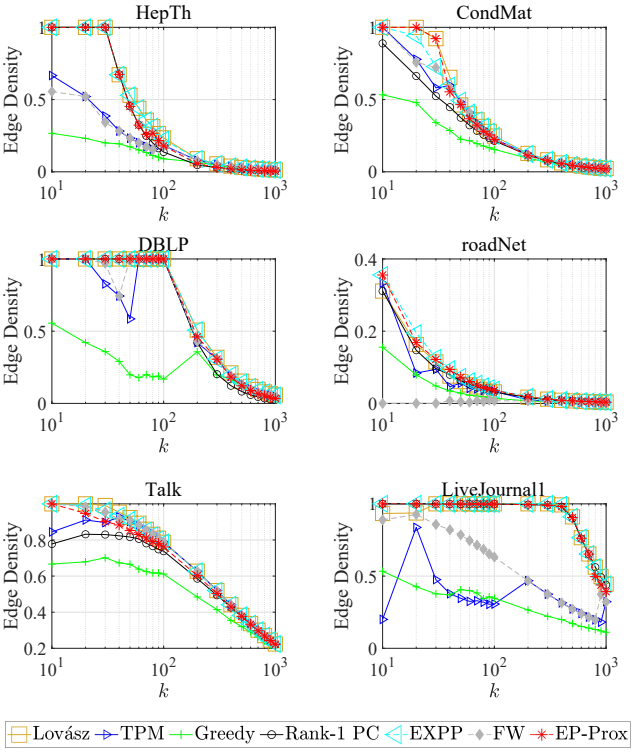

Figure 1: Edge density under different k for DkS problem.

Edge Density Performance: Figure 1 illustrates the density performance of different algorithms across k. Our proposed EP-Prox method achieves performance competitive with that of EXPP and the Lovász relaxation on all datasets. While Rank-1 PC, TPM, and FW can work well in some instances, they are less consistent. Greedy, in particular, often yields significantly lower edge density.

Runtime Comparison: Figure 2 reports the runtime of all the methods. Compared to the Lovász relaxation and EXPP, our EP-Prox method demonstrates significantly lower computational cost and is 2X faster across all datasets, thereby highlighting its scalability.

The Dk_1k_2BS Problem

We evaluate our method on several real-world bipartite networks obtained from the KONECT repository (Kunegis 2013). The selected graphs cover a wide range of sizes, and detailed statistics of the datasets used in our experiments are summarized in Table 2.

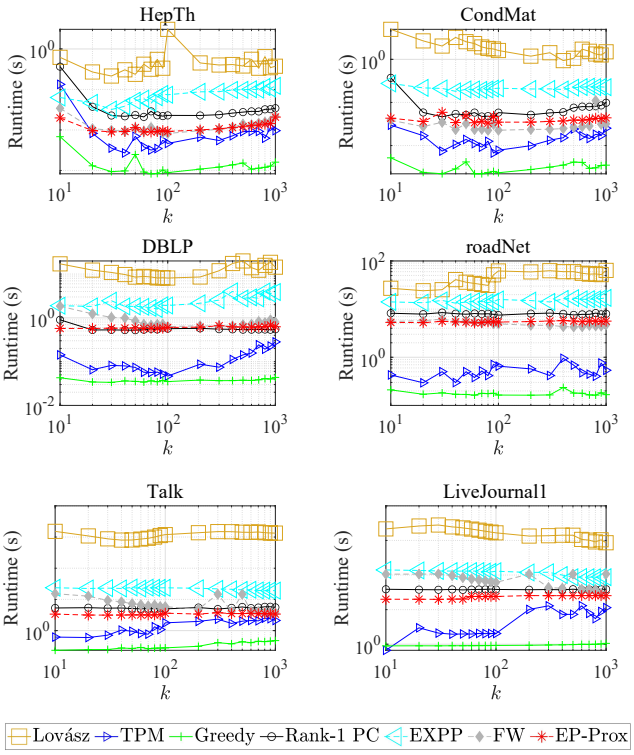

Figure 2: Runtime under different k for DkS problem.

Datasets	n_1	n_2	m
Actor movies	127K	383K	1.47M
Digg votes	139K	3,553	3M
Flickr	396K	103K	8.5M
Wikipedia edits (en)	8.1M	42.6M	572.6M

Table 2: Statistics of the bipartite datasets, where n_1, n_2 denote the sizes of the two node sets, and m is the number of edges.

To quantify the density of the extracted subgraph, we use the metric:

$$\text{edge density} = \boldsymbol{x}^\top \boldsymbol{B}\boldsymbol{y}/(k_1 k_2),$$

where $\boldsymbol{x} \in \mathcal{U}_{k_1}^{n_1}$ and $\boldsymbol{y} \in \mathcal{U}_{k_2}^{n_2}$ are the binary selection vectors returned by the respective methods. A higher edge density indicates a denser subgraph and thus better solution quality. We adapt the rank-1 PC and EXPP methods to solve the Dk_1k_2BS problem, and use them as baselines for comparison. This method follows the implementation in Algorithm 1, similar to DkS problem. In this test, we change the value of k_1 from 10 to 1000, and set $k_2 = 100$. The results in terms of edge density and runtime are shown in Figures 3 and 4. Additional details on the pre-processing steps, algorithm settings, and additional results for the case of $k_1 = k_2$ are provided in Part F of the supplementary material.

Edge Density and Runtime Comparison: From Figure 3, both EXPP and EP-Prox outperform Rank-1 PC, yielding denser subgraphs. Meanwhile, from Figure 4, EP-Prox is noticeably faster than EXPP (about an order of magnitude).

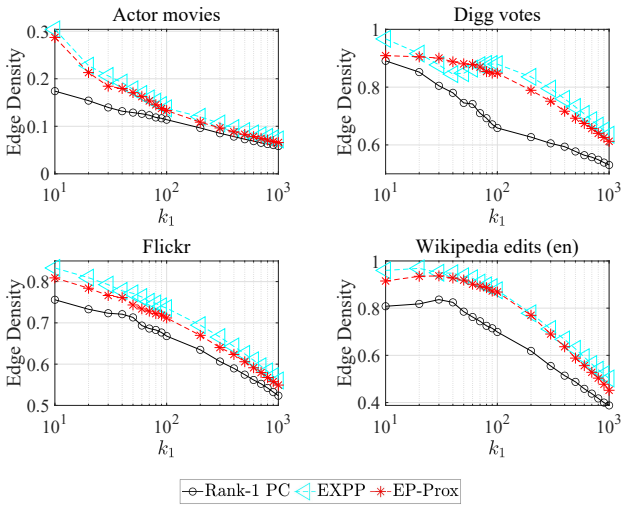

Figure 3: Edge density under different k_1 for Dk_1k_2BS.

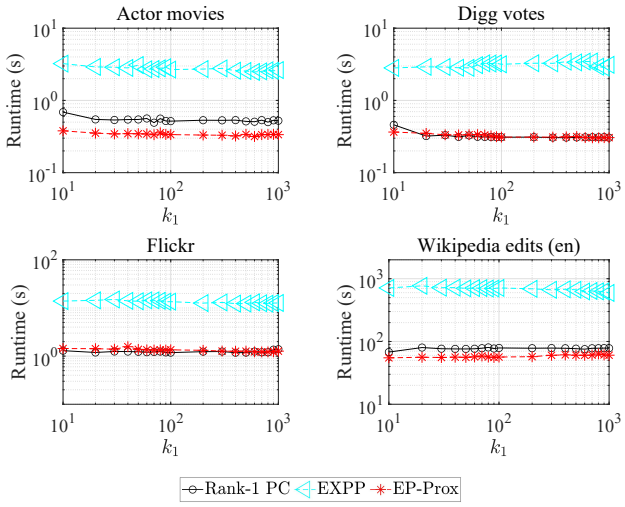

Figure 4: Runtime under different k_1 for Dk_1k_2BS.

Hence, EP-Prox attains a better performance-complexity trade-off.

Conclusion

We proposed a novel exact penalized formulation for the DkS problem based on the error bound principle, preserving both global and local optima. To solve it efficiently, we developed a non-convex proximal gradient method with low per-iteration cost and provided a convergence analysis. Extensive experiments on real-world large-scale graphs demonstrate that our approach achieves a good balance between the solution quality and the computation expense. Future work will explore the applicability of this framework for general cardinality-constrained optimization problems.

Acknowledgments

Ya Liu, Junbin Liu, and Wing-Kin Ma were supported by a General Research Fund (GRF) of Hong Kong Research Grant Council (RGC) under Project ID CUHK 14203721 and by a CUHK Direct Grant under Project ID 4055268. Aritra Konar was supported by KU Leuven Special Research Fund BOF/STG-22-040.

References

Andersen, R.; and Chellapilla, K. 2009. Finding dense subgraphs with size bounds. In *International Workshop on Algorithms and Models for the Web-Graph*, 25–37. Springer.

Angel, A.; Koudas, N.; Sarkas, N.; Srivastava, D.; Svendsen, M.; and Tirthapura, S. 2014. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *The VLDB Journal*, 23(2): 175–199.

Asahiro, Y.; Iwama, K.; Tamaki, H.; and Tokuyama, T. 2000. Greedily finding a dense subgraph. *Journal of Algorithms*, 34(2): 203–221.

Beck, A.; and Teboulle, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1): 183–202.

Bhaskara, A.; Charikar, M.; Chlamtac, E.; Feige, U.; and Vijayaraghavan, A. 2010. Detecting High Log-Densities: an $O(n^{1/4})$ Approximation for Densest k-Subgraph. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, 201–210. ACM.

Bhaskara, A.; Charikar, M.; Guruswami, V.; Vijayaraghavan, A.; and Zhou, Y. 2012. Polynomial integrality gaps for strong sdp relaxations of densest k-subgraph. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, 388–405. SIAM.

Bombina, P.; and Ames, B. 2020. Convex Optimization for the Densest Subgraph and Densest Submatrix Problems. In *SN Operations Research Forum*, volume 1, 1–24. Springer.

Boob, D.; Gao, Y.; Peng, R.; Sawlani, S.; Tsourakakis, C.; Wang, D.; and Wang, J. 2020. Flowless: Extracting densest subgraphs without flow computations. In *Proceedings of The Web Conference 2020*, 573–583.

Charikar, M. 2000. Greedy approximation algorithms for finding dense components in a graph. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, 84–95. Springer.

Chekuri, C.; Quanrud, K.; and Torres, M. R. 2022. Densest Subgraph: Supermodularity, Iterative Peeling, and Flow. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1531–1555. SIAM.

Chen, T.; and Tsourakakis, C. 2022. Antibenford subgraphs: Unsupervised anomaly detection in financial networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2762–2770.

Cui, Y.; and Pang, J.-S. 2021. *Modern Nonconvex Nondifferentiable Optimization*. SIAM.

Danisch, M.; Chan, T.-H. H.; and Sozio, M. 2017. Large scale density-friendly graph decomposition via convex programming. In *Proceedings of the 26th International Conference on World Wide Web*, 233–242.

Feige, U.; Peleg, D.; and Kortsarz, G. 2001. The dense k-subgraph problem. *Algorithmica*, 29(3): 410–421.

Feige, U.; Seltser, M.; et al. 1997. *On the Densest k-Subgraph Problem*. Weizmann Institute of Science.

Goldberg, A. V. 1984. *Finding a Maximum Density Subgraph*. Technical report, University of California Berkeley, CA.

Hager, W. W.; Phan, D. T.; and Zhu, J. 2016. Projection Algorithms for Nonconvex Minimization with Application to Sparse Principal Component Analysis. *Journal of Global Optimization*, 65: 657–676.

Harb, E.; Quanrud, K.; and Chekuri, C. 2022. Faster and scalable algorithms for densest subgraph and decomposition. *Advances in Neural Information Processing Systems*, 35: 26966–26979.

Harb, E.; Quanrud, K.; and Chekuri, C. 2023. Convergence to Lexicographically Optimal Base in a (Contra) Polymatroid and Applications to Densest Subgraph and Tree Packing. In *31st Annual European Symposium on Algorithms (ESA 2023)*, volume 274, 56. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Hooi, B.; Song, H. A.; Beutel, A.; Shah, N.; Shin, K.; and Faloutsos, C. 2016. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 895–904.

Karisch, S. E.; Rendl, F.; and Clausen, J. 2000. Solving graph bisection problems with semidefinite programming. *INFORMS Journal on Computing*, 12(3): 177–191.

Kawase, Y.; and Miyauchi, A. 2018. The densest subgraph problem with a convex/concave size function. *Algorithmica*, 80(12): 3461–3480.

Keys, K. L.; Zhou, H.; and Lange, K. 2019. Proximal distance algorithms: Theory and practice. *Journal of Machine Learning Research*, 20(66): 1–38.

Khuller, S.; and Saha, B. 2009. On finding dense subgraphs. In *International Colloquium on Automata, Languages, and Programming*, 597–608. Springer.

Konar, A.; and Sidiropoulos, N. D. 2021. Exploring the subgraph density-size trade-off via the Lovaśz extension. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 743–751.

Kunegis, J. 2013. Konect: the koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*, 1343–1350.

Lanciano, T.; Miyauchi, A.; Fazzone, A.; and Bonchi, F. 2024. A survey on the densest subgraph problem and its variants. *ACM Computing Surveys*, 56(8): 1–40.

Leskovec, J.; and Krevl, A. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

Li, J.; So, A. M.-C.; and Ma, W.-K. 2020. Understanding notions of stationarity in nonsmooth optimization: A guided tour of various constructions of subdifferential for nonsmooth functions. *IEEE Signal Processing Magazine*, 37(5): 18–31.

Li, X.; Liu, S.; Li, Z.; Han, X.; Shi, C.; Hooi, B.; Huang, H.; and Cheng, X. 2020. Flowscope: Spotting money laundering based on graphs. In *Proceedings of the AAAI Conference on Artificial aintelligence*, volume 34, 4731–4738.

Liu, J.; Liu, Y.; Ma, W.-K.; Shao, M.; and So, A. M.-C. 2024a. Extreme Point Pursuit—Part I: A Framework for Constant Modulus Optimization. *IEEE Transactions on Signal Processing*.

Liu, J.; Liu, Y.; Ma, W.-K.; Shao, M.; and So, A. M.-C. 2024b. Extreme Point Pursuit—Part II: Further Error Bound Analysis and Applications. *IEEE Transactions on Signal Processing*.

Liu, Y.; Liu, J.; and Ma, W.-K. 2024. Cardinality-constrained binary quadratic optimization via extreme point pursuit, with application to the densest k-subgraph problem. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 9631–9635. IEEE.

Lu, Q.; Sidiropoulos, N. D.; and Konar, A. 2025. Densest k-subgraph mining via a provably tight relaxation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 12291–12299.

Luo, W.; Ma, C.; Fang, Y.; and Lakshman, L. V. 2023. A Survey of Densest Subgraph Discovery on Large Graphs. ArXiv preprint arXiv:2306.07927, arXiv:2306.07927.

Luo, Z.-Q.; Pang, J.-S.; and Ralph, D. 1996. *Mathematical Programs with Equilibrium Constraints*. Cambridge University Press.

Manurangsi, P. 2017. Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 954–961.

Motzkin, T. S.; and Straus, E. G. 1965. Maxima for graphs and a new proof of a theorem of Turán. *Canadian Journal of Mathematics*, 17: 533–540.

Nesterov, Y. 2018. *Lectures on Convex Optimization*, volume 137. Springer.

Nguyen, T. D.; and Ene, A. 2024. Multiplicative Weights Update, Area Convexity and Random Coordinate Descent for Densest Subgraph Problems. In *International Conference on Machine Learning*, 37683–37706. PMLR.

Nocedal, J.; and Wright, S. J. 2006. *Numerical Optimization*. Springer.

Papailiopoulos, D.; Mitliagkas, I.; Dimakis, A.; and Caramanis, C. 2014. Finding dense subgraphs via low-rank bilinear optimization. In *International Conference on Machine Learning*, 1890–1898. PMLR.

Rockafellar, R. T.; and Wets, R. J.-B. 2009. *Variational Analysis*, volume 317. Springer.

Saha, B.; Hoch, A.; Khuller, S.; Raschid, L.; and Zhang, X.-N. 2010. Dense subgraphs with restrictions and applications to gene annotation graphs. In *Research in Computational Molecular Biology: 14th Annual International Conference, RECOMB 2010, Lisbon, Portugal, April 25-28, 2010. Proceedings 14*, 456–472. Springer.

Seidman, S. B. 1983. Network structure and minimum degree. *Social Networks*, 5(3): 269–287.

Shao, M.; and Ma, W.-K. 2020. Binary MIMO detection via homotopy optimization and its deep adaptation. *IEEE Transactions on Signal Processing*, 69: 781–796.

Shin, K.; Eliassi-Rad, T.; and Faloutsos, C. 2016. Corescope: Graph mining using k-core analysis—patterns, anomalies and algorithms. In *IEEE 16th International Conference on Data Mining (ICDM)*, 469–478. IEEE.

Sotirov, R. 2020. On Solving the Densest k-Subgraph Problem on Large Graphs. *Optimization Methods and Software*, 35(6): 1160–1178.

Tsourakakis, C.; Bonchi, F.; Gionis, A.; Gullo, F.; and Tsiarli, M. 2013. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 104–112.

Veldt, N.; Benson, A. R.; and Kleinberg, J. 2021. The generalized mean densest subgraph problem. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1604–1614.

Xu, Y.; and Yin, W. 2017. A globally convergent algorithm for nonconvex optimization based on block coordinate update. *Journal of Scientific Computing*, 72(2): 700–734.

Yuan, X.-T.; and Zhang, T. 2013. Truncated power method for sparse eigenvalue problems. *The Journal of Machine Learning Research*, 14(1): 899–925.

Zaslavskiy, M.; Bach, F.; and Vert, J.-P. 2008. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12): 2227–2242.