Efficient Anchored Densest Subgraph Discovery: Improved Time Complexity and Practical Performance

YINGLI ZHOU, The Chinese University of Hong Kong, Shenzhen, China
YOURAN SUN, The Chinese University of Hong Kong, Shenzhen, China
YIXIANG FANG*, The Chinese University of Hong Kong, Shenzhen, China

Densest subgraph discovery (DSD) is a fundamental research topic in network science and graph databases. As a typical variant of DSD, the anchored densest subgraph (ADS) problem aims to detect a densest subgraph that is anchored around user-defined seed vertices. The ADS problem has been shown to be very useful in some personalized applications, such as community search and recommendation. While the ADS problem is very useful, existing algorithms suffer from weak theoretical guarantees and perform poorly in practice. To address these issues, we first propose a novel approximation algorithm with improved time complexity, achieving the same accuracy as existing methods while requiring significantly fewer iterations. To further enhance practical performance, we introduce a graph reduction technique that localizes the ADS search to a much smaller subgraph, while still providing non-trivial theoretical guarantees. Building on this approximation, we also develop an efficient exact algorithm. We have performed an extensive empirical evaluation of our approaches on 12 real large datasets. The results show that our proposed algorithms are up to four orders of magnitude faster than the state-of-the-art.

CCS Concepts: • **Theory of computation → Algorithm design techniques**.

Additional Key Words and Phrases: densest subgraph, convex programming, graph density

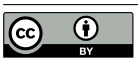
ACM Reference Format:
Yingli Zhou, Youran Sun, and Yixiang Fang. 2026. Efficient Anchored Densest Subgraph Discovery: Improved Time Complexity and Practical Performance. *Proc. ACM Manag. Data* 4, 3 (SIGMOD), Article 167 (June 2026), 27 pages. https://doi.org/10.1145/3802044

1 Introduction

Densest Subgraph Discovery (DSD) is a fundamental problem in graph data management, and has been studied for decades in the literature [2, 5, 14, 27, 48, 50, 59, 64, 66, 68, 80, 81]. The classical DSD problem aims to find the subgraph with maximum edge-density, or the number of edges over the number of vertices within the subgraph. This problem has also been extended to other definitions of density [49, 80, 81] and to different types of graphs [17, 50, 59]. Nevertheless, all of these DSD problems seek to find a single subgraph from the entire graph, and are thus referred to as *global DSD problems*. Such problems have found applications in many areas, such as community detection [26, 27, 46, 80], revealing fake followers [50], and identifying echo chambers and groups engaged in spreading misinformation [42]. While global DSD excels at identifying the single most dense region in a graph, it often fails to capture user-specific interests or localized structures that are still crucial in real-world applications. To address this, recent research has shifted toward *local DSD* (or

*Corresponding author.

Authors' Contact Information: Yingli Zhou, yinglizhou@link.cuhk.edu.cn, The Chinese University of Hong Kong, Shenzhen, Shenzhen, China; Youran Sun, youransun@link.cuhk.edu.cn, The Chinese University of Hong Kong, Shenzhen, Shenzhen, China; Yixiang Fang, The Chinese University of Hong Kong, Shenzhen, Shenzhen, China, fangyixiang@cuhk.edu.cn.



ACM 2836-6573/2026/6-ART167
https://doi.org/10.1145/3802044

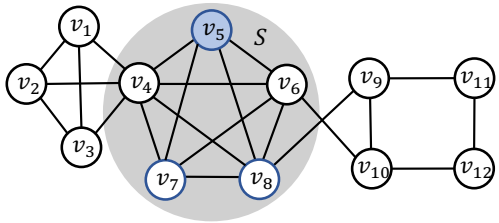


Fig. 1. An example of the anchored densest subgraph, in which A={v_5}, R={v_5, v_7, v_8}.

anchored densest subgraph, ADS) discovery, which seeks to find a densest subgraph (DS) anchored around user-defined seed vertices [20, 26, 44, 46, 51, 74, 76].

In this paper, we study efficient solutions for the ADS discovery [20, 26, 44, 46, 51, 74, 76]. The ADS problem provides a flexible framework for local DSD by imposing two critical, user-defined constraints, focused on two vertex sets: an anchor set A and a reference set R (where $A \subseteq R$). The anchor set A specifies vertices that must be included in the result, ensuring the discovered DS is centered around key seed vertices. In contrast, the reference set R restricts the search space, enforcing locality and maintaining contextual relevance of the resulting subgraph. The ADS problem has demonstrated superior effectiveness in identifying cohesive local communities that are centered on the anchor set. We refer readers to [21] for more details on different density measures. In light of this, we focus on the ADS problem with *NR-density* in this paper, that is, finding the subgraph with the highest NR-density. Given a graph $G = (V, E)$, where V and E denote the vertex set and edge set, respectively, for any vertex subset $S \subseteq V$, $G[S]$ denotes the *induced subgraph* on S, and $E(S)$ denotes the edge set of $G[S]$. For a vertex $u \in S$, $N(u, G[S]) = \{v \in S \mid (u, v) \in E\}$ is the neighbor set of u *within* $G[S]$. Given two vertex sets $A \subseteq R \subseteq V$ and any $S \subseteq V$, the *NR-density* [21, 74] of the induced subgraph $G[S]$ is:

$$\rho_R(G[S]) \;=\; \frac{2|E(S)| - \sum_{u \in S \setminus R} \big|N(u, G[S])\big|}{|S|}. \tag{1}$$

As shown in Fig. 1, given A={v_5}, and R={v_5, v_7, v_8}, the ADS is the subgraph induced by the vertex set $\{v_4, v_5, v_6, v_7, v_8\}$, whose NR-density is $\frac{20-8}{5}$=2.4, and there is no other subgraphs whose NR-density is larger than 2.4. The ADS problem remains highly relevant to graph data management and graph database systems. In particular, ADS has been widely used in fundamental tasks such as graph visualization, community detection (or search), and near-clique identification [26, 43, 46]. Moreover, modern graph database systems (e.g., Neo4j and Nebula) provide native community-oriented operators closely related to ADS discovery. Last but not least, the ADS problem has been found useful in many personalized applications [19, 20, 26, 44, 46, 74], to name a few:

- **Product recommendation** [20]. In e-commerce, the ADS problem can be applied to bundle recommendation. When a customer views a smartphone (the anchor set, A), all compatible accessories, such as cases, chargers, or earbuds, constitute the reference set, R. In this context, ADS corresponds to a dense subgraph with respect to the reference set R, representing a coherent item bundle that can be leveraged to enhance bundle recommendation performance, as shown in our case study in Section 7.3.
- **Community search** [44, 74]. In social and collaboration networks, communities can be naturally modeled as ADSs by using the anchor set (A) and reference set (R) as a guide. For example, on platforms such as GitHub, a key project maintainer can serve as the anchor set A, while all project contributors constitute the reference set R. In this setting, ADS denotes the core development team, a group of members who not only interact with the anchor but

also collaborate extensively with each other. Our experiments in Section 7.3 show that ADS is effective for revealing communities in the DBLP network.

- **Event organization** [20, 26, 46]. For example, when organizing an academic workshop, confirmed keynote speakers (e.g., "Jim Gray") can be treated as the anchor set (A), while researchers who closely collaborate with the anchors constitute the reference set (R). Then, ADS corresponds to a recommended list of invitees—researchers who maintain strong co-authorship or collaboration ties with the anchors and with one another. We provide a case study in Section 7.3 to demonstrate the effectiveness of this formulation.

• **Prior works.** In such personalized applications, ADS must be repeatedly computed for many different anchored and reference sets on large-scale graphs. In such scenarios, ADS discovery often becomes an efficiency bottleneck. In the literature, a few exact and approximation algorithms have been developed to solve the ADS problem. The exact algorithms are often based on maximum flow [20, 26] and convex programming [74]. The approximation algorithms are based on peeling [14, 39, 40, 47, 50], and convex programming [21, 37, 74]. Among these algorithms, the state-of-the-art (SOTA) approximation algorithm is FDP [74], a convex programming-based method that relies on optimization solvers to efficiently compute solutions. Building upon it, the SOTA exact algorithm FDPE [74] first uses FDP to obtain a high-quality approximate solution and then verifies its optimality via the maximum flow algorithm [56]. While the above methods demonstrate promising results, based on our careful analysis, we observe that they exhibit two key limitations:

• **Limitation 1: Weak theoretical convergence guarantee.** The theoretical convergence of FDP to a $(1+\epsilon)$-approximation solution has not been formally analyzed [74]. To fill this gap, in this paper, we establish the first such guarantee, showing that FDP requires $O\left(\frac{\Delta(G)\cdot(\mathrm{vol}(R)+|A|\Delta(G)^2)}{\epsilon^2}\right)$ iterations, where $\mathrm{vol}(R)$ is the volume of vertices in R and $\Delta(G)$ is the maximum degree. This weak bound implies that achieving a near-optimal ADS requires a large number of iterations. For instance, on the EF dataset (4,039 vertices, 88,234 edges), FDP requires over 10 million iterations to achieve a 1.1-approximation (Table 2). Consequently, the slow convergence of FDP becomes a significant performance bottleneck for FDPE.

• **Limitation 2: Inefficiently large search space.** Current algorithms like FDP and FDPE operate on the entire graph, an approach that is highly inefficient given the localized nature of the solution. Our empirical analysis on real-world graphs reveals a stark mismatch: the resulting ADS is minuscule, consistently comprising no more than 0.002% of the total vertices (Table 3). Because the density metric inherently filters out low-degree vertices, the search space can be drastically reduced. This motivates our pre-processing approach to first isolate a much smaller candidate subgraph that is guaranteed to contain the solution. As a result, existing methods do not scale to large real-world graphs, which further motivates our work on efficient ADS algorithms.

• **Our technical contributions.** To address Limitation 1, we propose PASTA, a novel algorithm with a rigorous $(1+\epsilon)$-approximation guarantee for $0 < \epsilon < 1$. Notably, to obtain a $(1+\epsilon)$-approximation solution, PASTA only takes $O\left(\frac{\sqrt{|A|}\cdot\mathrm{vol}(R)^{\frac{1}{4}}\cdot\Delta(G)^{\frac{3}{4}}\cdot\sqrt{(\mathrm{vol}(R)+|A|\Delta(G))}}{\epsilon}\right)$ iterations rather than $O\left(\frac{\Delta(G)\cdot(\mathrm{vol}(R)+|A|\Delta(G)^2)}{\epsilon^2}\right)$, which is needed by the existing algorithms. As a result, PASTA **reduces the total runtime** (the product of the iteration count and per-iteration complexity) by $O\left(\frac{\mathrm{vol}(R)^{\frac{3}{4}}\cdot\Delta(G)^{\frac{5}{4}}}{\epsilon}\right)$ over FDP, thereby enabling efficient ADS discovery for any ϵ on large graphs. Specifically, in PASTA, we propose a new quadratic programming (QP) formulation for the ADS problem and theoretically prove that its optimal solution corresponds to the ADS. Building on this, we design a projected gradient descent (PGD) [28]-based approximation algorithm for the ADS

problem, which iteratively optimizes the QP formulation. In each iteration, PASTA first updates all edge weights using their gradients and projected gradients, and then, for each vertex, aggregates the weights of its incident edges to update its own weight. Finally, the vertices with the highest weights are selected to form the ADS.

To address **Limitation 2**, we propose an effective graph reduction technique based on a novel cohesive subgraph structure, the k-$r$core, which we formally define in Section 5.1. Extending the classic k-core (the maximal subgraph where every vertex has a degree of at least k) [8], we theoretically prove that the ADS is guaranteed to reside within a specific k-$r$core. This key result allows us to safely prune the graph; we can restrict the search space to a k-$r$core, which is significantly smaller than the entire graph, thereby dramatically improving efficiency.

Following the insights above, we develop an exact algorithm, named PASTA-Exact, which first invokes PASTA to obtain a high-quality approximate solution and then verifies its optimality through the maximum-flow algorithm [57]. Extensive experimental evaluations on 12 real-world large graphs show that PASTA achieves both higher efficiency and scalability than the existing ADS algorithms on all datasets. Particularly, our approximation and exact algorithms are up to four and one order(s) of magnitude faster than SOTA algorithms, respectively. We have released the source codes and datasets of our work: https://github.com/YouranSun/AnchoredDS.

In summary, our principal contributions are as follows.

- We develop an efficient ADS algorithm based on the gradient projection to break the theoretical bottleneck of existing ADS algorithms, achieving a better theoretical guarantee of the convergence rate.
- To localize the ADS into a small subgraph, we propose an effective graph reduction technique with a non-trivial theoretical guarantee. We further propose some useful optimization techniques to boost the efficiency and design an exact algorithm.
- We conduct experiments on 12 real-world large graphs to demonstrate the efficiency and scalability of our algorithm.

Outline. We introduce the ADS problem in Section 2. Section 3 analyzes the limitations of state-of-the-art algorithms. We introduce our PASTA and PASTA-Exact algorithms in Section 6. The experimental results are reported in Section 7. We review the related works in Section 8 and conclude in Section 9. For lack of space, all detailed proofs in this paper are included in our supplementary material [84], and we only show the proof sketches in our manuscript.

2 Preliminaries

In this section, we first present the formal definition of the ADS problem and then introduce its QP formulations.

2.1 Problem definition

Table 1. Notations and meanings.

Notation	Meaning
$G = (V, E)$	a graph with vertex set V and edge set E
$G[H]$	the subgraph of G induced by vertices in H
$N(v, G)$	The set of neighbors of v in G
$\rho_R(S)$	The NR-density of subgraph induced by vertex set S
$d_R(v, G)$	The R-degree of v in G
ρ_R^*	The NR-density of the anchored densest subgraph of G
$\underline{\rho_R}$	The lower bound of optimal NR-density ρ_R^*

In this paper, we consider an unweighted, and undirected graph G=(V, E), where V and E are the sets of vertices and edges in the graph, respectively. We denote by $n = |V|$ and $m = |E|$ ($m > n$) the numbers of vertices and edges in G, respectively. The subgraph induced by vertices in H, denoted by $G[H]$=$(V(H), E(H))$. For each vertex $v \in V$, we use $N(v, G)$ to denote its set of neighbors in G. Next, we formally present the definition of NR-density [74] and the problem of Anchored Densest Subgraph discovery, or ADS problem. We summarize the frequently used notations in Table 1.

Definition 2.1 (NR-density [21, 74]). Given a graph G=(V, E) and a vertex set $R \subseteq V$, the NR-density of the subgraph induced by the vertex set $S \subseteq V$ is:

$$\rho_R(G[S]) = \frac{2|E(S)| - \sum_{u \in S \setminus R} |N(u, G[S])|}{|S|}. \tag{2}$$

Definition 2.2 (Anchored Densest Subgraph [20, 21, 74]). Given a graph G=(V, E), and two vertex sets $A, R \subseteq V$, an *anchored densest subgraph* (ADS) of G is a subgraph of G that contains A, whose NR-density is the highest among all possible subgraphs of G. We use $\mathcal{D}$ to denote the ADS of G, and ρ_R^*=$\rho_R(\mathcal{D})$ to denote the NR-density of the ADS.

Example 1. *In the graph G shown in Fig. 1, let $A = \{v_5\}$, $R = \{v_5, v_7, v_8\}$, and S=$\{v_4, v_5, v_6, v_7, v_8\}$. We have $|E(S)| = 10$, $|S| = 5$, and $\sum_{u \in S \setminus R} |N(u, G[S])| = |N(v_4, G[S])| + |N(v_6, G[S])| = 8$. Therefore, $\rho_R(G[S]) = \frac{2 \times 10 - 8}{5} = 2.4$. Clearly, $G[S]$ is the ADS since no other subgraph has a higher NR-density.*

In this work, we primarily study the $(1 + \epsilon)$-approximation solution, where $0 < \epsilon \leq 1$. The approximation ratio of an algorithm is defined as the density of ADS over that of the returned subgraph, which is at least 1.0. Furthermore, in this paper, we also design efficient exact algorithms (i.e., ϵ=0) for the ADS problem.

Next, we define the problem that we mainly study:

Problem 1 (ADS problem [20, 21, 26, 46, 74]). *Given a graph G=(V, E), and two vertex sets $A, R \subseteq V$, return an ADS $\mathcal{D}$ of G.*

Note that a graph may contain multiple ADSs, and our paper focuses on finding one such ADS.

In the literature, localized density measures can be categorized into two groups [21]: 1) *strongly local*, and 2) *non–strongly local*, where the former exhibits strict locality with respect to the reference set R and can be computed more efficiently than the latter. We choose NR-density, since it belongs to the class of strongly local densities. In addition, through our initial investigation, we believe that the key ideas in our proposed algorithm can also be adapted to other strongly local density measures. In Section 7.3, we demonstrate that ADS under NR-density consistently outperforms that under R-density across two downstream applications, namely community search and bundle recommendation.

2.2 The QP formulation of the ADS problem

We first introduce the quadratic programming (QP) formulation of the ADS problem [74], which serves as the foundation for the SOTA ADS algorithms. Specifically, for a given vertex set $A \subseteq V$:

$$
\begin{aligned}
\mathrm{QP}(G,A):\ \min & \sum_{u\in V}\left(r(u)+\beta(u)\right)^2 & \\
\text{s.t.}\ & \sum_{(u,v)\in E}\alpha_{u,v}=r(u), & \forall u\in V\\
& \alpha_{u,v}+\alpha_{v,u}=w_{u,v}, & \forall (u,v)\in E\\
& \beta(u)=0, & \forall u\notin A\\
& r(u)+\beta(u)=\max_{v\in V}\left(r(v)+\beta(v)\right), & \forall u\in A\\
& \alpha_{u,v}\geq 0,\ \alpha_{v,u}\geq 0, & \forall (u,v)\in E
\end{aligned}
$$

where $\alpha_{u,v}$ indicates the weight assigned to u and v from an edge (u,v), and for an edge $(u,v)\in E$, $w_{u,v}=\mathbb{1}[u\in R]+\mathbb{1}[v\in R]$. Clearly, $w_{u,v}$ reflects whether the endpoints of an edge belong to R. The intuition behind this design is that edges incident to nodes in R are typically assigned greater weights. Here, we define two vectors $\boldsymbol{r}\in\mathbb{R}^n$, and $\boldsymbol{\beta}\in\mathbb{R}^n$:

$$
\boldsymbol{r}=\begin{bmatrix} r(v_1) & \cdots & r(v_n)\end{bmatrix}\quad \boldsymbol{\beta}=\begin{bmatrix}\beta(v_1) & \cdots & \beta(v_n)\end{bmatrix}.
$$

Indeed, the optimal solution of $\mathrm{QP}(G,A)$ (denoted by $\mathtt{OPT}(\mathrm{QP})$) exactly corresponds to the optimal ADS density, i.e., $\mathtt{OPT}(QP)=\rho_R^*$, which forms the core of the baseline method FDP. Intuitively, $\mathrm{QP}(G,A)$ distributes edge weights to vertices such that the received weights are as balanced as possible, and any feasible solution corresponds to a subgraph that necessarily contains the anchored set A [74]. For a vertex v, the quantity $r(v)+\beta(v)$ represents its overall contribution to the ADS: larger values indicate a higher likelihood of being included. By Fujishige's theorem [29], in the optimal solution all vertices in the ADS share the same value, i.e., for the optimal ADS $G[S]$ and any $v\in S$, $r(v)+\beta(v)=\rho_R^*$, while vertices outside S have strictly smaller values. This property directly explains why we select vertices with the largest values of $r+\beta$ as the candidate ADS, a strategy commonly adopted in densest subgraph–related problems. The variables r and β play complementary roles: $r(v)$ captures the intrinsic contribution of v to the NR-density objective, whereas $\beta(v)$ serves as an adjustment variable to enforce the anchoring constraint. In particular, $\beta^*(u)=0$ for any $u\notin A$, meaning non-anchored vertices are selected purely based on $r(u)$. We defer the full details to our supplementary material.

2.3 The first-order optimization problem

First-order methods are widely used for solving convex optimization problems due to their simplicity and low cost [33, 37, 54, 80]. These methods rely solely on gradient information. A standard form of constrained convex optimization is: $\min_{x\in C} f(x)$, where f is a differentiable convex function, and C is a closed convex set.

One of the most fundamental first-order algorithms is *Projected Gradient Descent (PGD)* [53]. At each iteration, PGD performs a gradient descent step followed by a projection onto the feasible region:

$$
x_{t+1}=\mathtt{Prox}_C\big(x_t-\eta_t\cdot\nabla f(x_t)\big),
$$

where $\eta_t>0$ is the step size, $\nabla f(x_t)$ is the gradient of $f(x_t)$ and $\mathtt{Prox}_C(\cdot)$ denotes the Euclidean projection onto C. Given a point $z\in\mathbb{R}^n$, the Euclidean projection onto a convex set C is defined as:

$$
\mathtt{Prox}_C(z)=\arg\min_{x\in C}\|x-z\|_2^2.
$$

3 Analysis of SOTA Algorithms

In this section, we review the SOTA approximation and exact algorithms, both of which rely on convex programming via `Frank-Wolfe`, and then analyze their limitations to motivate our work.

3.1 Review of the state-of-the-art methods

The state-of-the-art (SOTA) algorithms for the ADS problem are built upon the well-known `Frank-Wolfe` algorithm [37], which serves as an efficient first-order solver for the QP(G, A) formulation described in Section 2.2.

Algorithm 1: `FDP` and `FPDE` [74]

input : A graph G=(V, E), two sets R, A, an integer T, and ϵ
output: An approximate/exact ADS $\mathcal{D}$

1 f $\leftarrow$`False`, $\mathcal{D} \leftarrow \emptyset$;
2 **foreach** $(u, v) \in E$ **do** $w_{u,v} \leftarrow \mathbb{1}[u \in R] + \mathbb{1}[v \in R]$;
3 **foreach** $u \in V$ **do** $r(u) \leftarrow 0$; $\beta(u) \leftarrow 0$;
4 **repeat**
5 $\quad \boldsymbol{r}, \boldsymbol{\beta} \leftarrow$ `Frank-Wolfe-ADS`(G, A, T);
$\quad$ `// Extract the ADS`
6 $\quad$ sort all $v \in V$ by weight $r(v) + \beta(v)$;
7 $\quad s^* \leftarrow \arg\max_{1 \le i \le n} \rho_R(V_i)$, where V_i is the top-i vertices in V;
$\quad$ `//` $\epsilon = 0$: `FDPE;` $\epsilon > 0$: `FDP`
8 $\quad$ **if** $G[V_{s^*}]$ *satisfies* $(1+\epsilon)$*-approx. ratio* **then**
9 $\quad\quad \mathcal{D} \leftarrow G[V_{s^*}]$; $f \leftarrow$ `True`;
10 **until** f=`True`;
11 **return** $\mathcal{D}$;

12 **Function** `Frank-Wolfe-ADS`(G, A, T)**:**
13 $\quad$ **foreach** $t = 1, 2, \cdots, T$ **do**
14 $\quad\quad$ **foreach** $e = (u, v) \in E$ *with* $w_{u,v} > 0$ **do**
15 $\quad\quad\quad$ **if** $r(u) < r(v)$ **then** $r(u) \leftarrow r(u) + w_{u,v}$;
16 $\quad\quad\quad$ **else if** $r(u) > r(v)$ **then** $r(v) \leftarrow r(v) + w_{u,v}$;
17 $\quad\quad\quad$ **else**
18 $\quad\quad\quad\quad r(u) \leftarrow r(u) + \frac{w_{u,v}}{2}$;
19 $\quad\quad\quad\quad r(v) \leftarrow r(v) + \frac{w_{u,v}}{2}$;
20 $\quad\quad \tau \leftarrow \max_{u \in V} (r(u) + \beta(u))$;
21 $\quad\quad$ **foreach** $u \in A$ **do** $\beta(u) \leftarrow \tau - r(u)$;
22 $\quad$ **return** $\boldsymbol{r}, \boldsymbol{\beta}$;

The authors in [74] propose an approximation algorithm, `FDP`, and an exact algorithm, `FDPE`, as shown in Algorithm 1. This algorithm iteratively linearizes the objective function at the current solution and moves towards its minimizer. Specifically, it takes as input a graph G, two sets R, A, and the number of iterations T, and outputs the approximate/exact ADS $\mathcal{D}$. The algorithm begins by initializing $w_{u,v}$ for each edge $(u, v) \in E$ as well as the vectors $\boldsymbol{r}$ and $\boldsymbol{\beta}$ (lines 1–3). It then enters a do-while loop to find an approximate or exact ADS (lines 4–11). Specifically, it first runs `Frank-Wolfe-ADS` for T iterations to update $\boldsymbol{r}$, and $\boldsymbol{\beta}$ (lines 12–22). In each iteration, for every edge $(u, v) \in E$, the algorithm attempts to assign its weight $w_{u,v}$ to the endpoint vertex with the smallest weight (lines 14–19). Afterwards, it computes the maximum value of $r(u) + \beta(u)$ over all vertices

Table 2. # of iterations of FDP and PASTA ($\epsilon = 0.01$).

Dataset	FDP	PASTA	Reduction
LJ	≥1,048,576	128	8,192×
SU	≥33,554,432	256	131,072×
UK	≥33,554,432	128	262,144×
IT	≥4,194,304	256	16,384×
SK	≥65,536	16	4,096×

Table 3. The ratio of the number of vertices in the ADS to that of the whole graph.

Datasets	**# of vertices**	$\lvert R\rvert < 100$		$\lvert R\rvert \geq 100$	
		$\lvert\mathcal{D}\rvert$	**ratio**	$\lvert\mathcal{D}\rvert$	**ratio**
YT	3,223,585	89	0.002%	58	0.001%
CP	3,774,768	34	0.0009%	32	0.0009%
LJ	4,847,571	40	0.0008%	61	0.001%
SU	58,790,782	50	0.0001%	57	0.0001%
UK	18,520,486	74	0.0003%	40	0.0002%

$u \in V$, denoted as τ (line 20). Finally, the algorithm updates $\beta(u)$ for all vertices $u \in A$ (line 21). Intuitively, vertices with higher weights are more likely to be included in the ADS $\mathcal{D}$, since they are incident to more edges. Thus, the subgraph induced by the highest-weight vertices with the largest density is considered an approximate ADS of G (lines 6–7). This process, also known as PAVA, is widely used in existing DSD algorithms [46, 81]. Finally, the algorithm verifies whether $G[V_{s^*}]$ satisfies the $(1+\epsilon)$-approximation criterion. If so, the algorithm terminates and returns $G[V_{s^*}]$ as the ADS (lines 8-9); otherwise, it repeats the above process until the criterion is met.

3.2 In-depth analysis of the SOTAs

To motivate our work, we first conduct an in-depth analysis of the leading ADS solution, FDP. While FDP is based on the standard Frank-Wolfe framework (Algorithm 1), a careful review of the literature reveals a critical gap: its theoretical convergence rate has not been formally established [74]. This omission makes it difficult to understand the fundamental performance characteristics and limitations of the algorithm. To address this, we provide the first formal analysis of FDP's convergence, summarized in the following theorem.

Theorem 3.1. *Given a graph $G = (V, E)$, and two vertex sets $R, A \subseteq V$. In* FDP, *when $T > O\left(\frac{\Delta(G)(\mathrm{vol(R)} + \mathrm{|A|}\Delta(\mathrm{G})^2)}{\epsilon^2}\right)$, the following convergence guarantee holds: $\|\boldsymbol{r} + \boldsymbol{\beta}\|_\infty - \rho_R^* \leq \epsilon$, where* vol(R) *is the sum of degrees in G for all vertices in the set R, and $\Delta(G)$ denotes the maximum degree of graph G.*

The above theorem guarantees that FDP is able to find a $(1+\epsilon)$-approximation solution after $O\left(\frac{\Delta(G)\cdot(\mathrm{vol(R)}+\mathrm{|A|}\Delta(\mathrm{G})^2)}{\epsilon^2}\right)$ iterations. Due to the space limitations, we only present the proof sketch of Theorem 3.1 here, and leave more details in our supplementary material [84].

Proof sketch. The update rule of FDP can be expressed as a standard Frank-Wolfe update with learning rate $\gamma_t = \frac{1}{t}$ for minimizing the function $f(z) = \|z\|^2$ over the feasible set C. Our proof proceeds in two steps: (1) we first derive a tight bound for the curvature constant ξ of the objective function $f(z)$, showing $\xi \leq 8(\Delta(G)+1)\cdot(|A|\cdot\Delta(G)^2 + \mathrm{vol}(R))$; (2) with this curvature

bound, we apply the standard convergence result for `Frank-Wolfe` from [37] to establish the final guarantee. □

Limitations: As analyzed above and in Section 1, the core limitations of baselines lie in: 1) the poor convergence rate, as shown in Theorem 3.1, and this theoretical limitation is also reflected in practice, as shown in Table 2. On the SU and UK datasets, `FDP` requires more than 300M iterations to find a 1.01-approximation solution. 2) the huge search space, since the existing algorithms must discover ADS in the whole graph. An interesting fact is that on most real-world graphs, the ratio of the number of vertices in the ADS comprises no more than 0.002% of the total vertices in the original graph, as shown in Table 3. This is mainly because the ADS problem aims to maximize the anchored NR-density where both R and A capture vertices in a small local region. This motivates us to design a new algorithm to address the above issues.

Takeaway: We note that the authors in [74] do not provide a theoretical analysis on the convergence rate of Algorithm 1, and only define $\mathrm{QP}(G, A)$ and prove that its optimal solution corresponds to the ADS. Our work is the first to fill this gap by providing a rigorous theoretical analysis of the convergence rate of the `Frank-Wolfe`-based ADS algorithm.

4 PGD-based ADS algorithm

In this section, we first propose a new QP formulation for the ADS problem, and based on this formulation, we propose a PGD-based ADS algorithm.

4.1 A new QP formulation for the ADS problem

In this subsection, we propose a new QP formulation, $\mathrm{QP}'(G, A)$ of the ADS problem, which keeps the same optimality of $\mathrm{QP}(G, A)$ in Section 2.2.

$$
\begin{aligned}
\mathrm{QP}'(G,A): \quad \min \quad & |A| \cdot \left(\max_{u \in A} r(u)\right)^2 + \sum_{u \in V \setminus A} r(u)^2 & \\
\text{s.t.} \quad & \sum_{(u,v) \in E} \alpha_{u,v} = r(u), & \forall u \in V \\
& \alpha_{u,v} \geq 0, & \forall (u,v) \in E \\
& \alpha_{u,v} + \alpha_{v,u} = w_{u,v}, & \forall (u,v) \in E
\end{aligned}
$$

Compared to $\mathrm{QP}(G, A)$, a notable feature of $\mathrm{QP}'(G, A)$ is that it has only one variable in the objective, making it easier to solve. Based on the vector $\boldsymbol{\alpha} \in \mathbb{R}^{2m}$, we define the function $f(\alpha)$ as:

$$
f(\alpha) = |A| \cdot \left(\max_{u \in V} \sum_{v \in N(u,G)} \alpha_{u,v}\right)^2 + \sum_{u \in V \setminus A} \left(\sum_{v \in N(u,G)} \alpha_{u,v}\right)^2,
$$

which is simply the objective function of $\mathrm{QP}'(G, A)$ rewritten in terms of α. In other words, the goal of the ADS problem is to minimize the $f(\alpha)$. In addition, we also introduce an indicator function $h(\alpha)$:

$$
h(\alpha) = \begin{cases} 0, \text{ if } \alpha_{u,v} + \alpha_{v,u} = w_{u,v} \text{ and } \alpha_{u,v}, \alpha_{v,u} \geq 0 \quad \forall e = (u,v) \in E \\ +\infty, \text{otherwise} \end{cases}
$$

Then, the objective function $\mathrm{QP}'(G, A)$ can be rewritten as minimizing the following unconstrained function:

$$\underbrace{f(\alpha)}_{\texttt{Smooth, Lipschitz-continuous}} + \underbrace{h(\alpha)}_{\texttt{Non-smooth}} . \tag{3}$$

We note that Equation (3) obeys the standard composite convex form: $\min_{x \in C} f(x) + h(x)$, where C is a closed convex set, $f(x)$ is a smooth convex function with a Lipschitz-continuous gradient and $h(x)$ is a non-smooth convex regularization term [53].

In the following, we demonstrate: (1) the optimal solution of Equation (3) is equivalent to $\texttt{OPT}(\mathrm{QP}(G, A))$; (2) how to design the PGD-based method to optimize Equation (3).

Theorem 4.1. *Given a graph $G = (V, E)$ and two vertex sets $R, A \subseteq V$, we have $\texttt{OPT}(QP(G, A)) = \texttt{OPT}(QP'(G, A))$.*

Proof sketch. We prove that $\mathrm{OPT}(\mathrm{QP}(G, A)) = \mathrm{OPT}(\mathrm{QP}'(G, A))$ by showing the inequality in both directions.

(i) $\mathrm{OPT}(\mathrm{QP}(G, A)) \geq \mathrm{OPT}(\mathrm{QP}'(G, A))$. Let $(\mathbf{r}^*, \boldsymbol{\beta}^*, \boldsymbol{\alpha}^*)$ be an optimal solution to $\mathrm{QP}(G, A)$. This implies that $(\mathbf{r}^*, \boldsymbol{\alpha}^*)$ is a feasible solution for $\mathrm{QP}'(G, A)$.

By the definition of optimality, the optimal value of $\mathrm{QP}'(G, A)$ is no greater than the objective value of any of its feasible solutions. Therefore:

$$\mathrm{OPT}(\mathrm{QP}'(G, A)) \leq \sum_{v \in V} (r^*(v))^2$$

Also, assuming $\beta^*(v) \geq 0$, it follows that $(r^*(v) + \beta^*(v))^2 \geq (r^*(v))^2$. Thus:

$$\mathrm{OPT}(\mathrm{QP}(G, A)) = \sum_{v \in V} (r^*(v) + \beta^*(v))^2 \geq \sum_{v \in V} (r^*(v))^2$$

Combining these, we get $\mathrm{OPT}(\mathrm{QP}(G, A)) \geq \mathrm{OPT}(\mathrm{QP}'(G, A))$.

(ii) $\mathrm{OPT}(\mathrm{QP}(G, A)) \leq \mathrm{OPT}(\mathrm{QP}'(G, A))$. Let $(\mathbf{r}^*, \boldsymbol{\alpha}^*)$ be an optimal solution to $\mathrm{QP}'(G, A)$. Based on the convexity of the objective function, we can assume that an optimal solution exists where $r^*(u)$ is constant for all $u \in A$, and this value is the maximum over all vertices. We construct a feasible solution for $\mathrm{QP}(G, A)$ based on this specific optimal solution $(\mathbf{r}^*, \boldsymbol{\alpha}^*)$. Let $\tau^* = \max_{v \in V} r^*(v)$, and define $\boldsymbol{\beta}^* = \mathbf{0}$ (the zero vector).

We verify the feasibility of $(\mathbf{r}^*, \boldsymbol{\beta}^*, \boldsymbol{\alpha}^*)$ for $\mathrm{QP}(G, A)$ with parameter τ^*:

- For all $u \in A$, we have $r^*(u) + \beta^*(u) = \tau^* + 0 = \tau^*$. The constraint is satisfied.
- For all $v \in V \setminus A$, we have $r^*(v) + \beta^*(v) = r^*(v) + 0 \leq \tau^*$. This constraint is also satisfied.

Thus, $(\mathbf{r}^*, \mathbf{0}, \boldsymbol{\alpha}^*)$ is a feasible solution for $\mathrm{QP}(G, A)$. Its objective function value is:

$$\sum_{v \in V} (r^*(v) + \beta^*(v))^2 = \sum_{v \in V} (r^*(v) + 0)^2 = \sum_{v \in V} (r^*(v))^2 = \mathrm{OPT}(\mathrm{QP}'(G, A))$$

Since we have found a feasible solution for $\mathrm{QP}(G, A)$ whose objective value equals $\mathrm{OPT}(\mathrm{QP}'(G, A))$, the optimal value of $\mathrm{QP}(G, A)$ must be less than or equal to this value. That is, we have $\mathrm{OPT}(\mathrm{QP}(G, A)) \leq \mathrm{OPT}(\mathrm{QP}'(G, A))$. □

Based on the above theorem, we establish that $\texttt{OPT}(QP(G, A)) = \texttt{OPT}(QP'(G, A)) = \rho^*_R$, and the ADS extracted based on the optimal solution of $\mathrm{QP}'(G, A)$ also contains the anchor set A.

4.2 Our PGD-based algorithm

In this subsection, we present the design of our PGD-based algorithm for solving the ADS problem. As discussed earlier, the ADS problem can be reduced to optimizing Equation (3). Although numerous PGD solvers have been proposed in the literature for general convex optimization, to the best of our knowledge, none have been explored in the context of the ADS problem.

We remark that applying the PGD solver for the ADS problem is not trivial; two key challenges need to be addressed: **Challenge 1: How to compute the gradient of function $f(\alpha)$?** and **Challenge 2: How to compute the proximal gradient of $h(\alpha)$?** To address **Challenge 1**, we derive the closed-form gradient of the smooth function $f(\alpha)$, which yields a $2m$-dimensional gradient vector $\nabla f(\alpha)$. For each edge $(u,v) \in E$, the corresponding entry in the gradient is given by:

$$\nabla f(\alpha)_{(u,v)} = 2\left(\mathbb{1}[u \notin A]r(u) + |A| \cdot \left(\max_{w \in V} r(w)\right) \cdot g_{u,v}\right),$$

where $g_{u,v}$ denotes the sub-gradient of the term $\max_{w \in V} r(w)$ w.r.t. $\alpha_{u,v}$, i.e., $g_{u,v} = \frac{\partial(\max_{w \in V} r(w))}{\alpha_{u,v}} =$

$$g_{u,v} = \begin{cases} \frac{1}{|\Psi|}, & u \in \Psi, \\ 0, & \text{otherwise}, \end{cases}$$

where $\Psi = \{u \mid u \in V \wedge r(u) = \max_{w \in V} r(w)\}$.

Algorithm 2: PGD-ADS

input : Graph $G=(V,E)$, two vertex sets R, A and an integer T
output: An approximate ADS $\mathcal{D}$

foreach $(u,v) \in E$ **do** $\alpha_{u,v}^{(0)} \leftarrow \frac{w_{u,v}}{2}$; $\alpha_{v,u}^{(0)} \leftarrow \frac{w_{v,u}}{2}$;
foreach $u \in V$ **do** $r^{(0)}(u) = \sum_{v \in N(u,G)} \alpha_{u,v}^{(0)}$;
$\eta \leftarrow \frac{1}{2 \cdot (|A|+1) \cdot \sqrt{\Delta(G)\mathrm{vol}(R)}}$; ▷ Learning rate
for $t = 1, 2, \cdots, T$ **do**
  for $(u,v) \in E$ **do**
    $\alpha_{u,v}^{(t)} \leftarrow \alpha_{u,v}^{(t)} - \eta \cdot \nabla f(\alpha)_{u,v}$;
    $\alpha_{v,u}^{(t)} \leftarrow \alpha_{v,u}^{(t)} - \eta \cdot \nabla f(\alpha)_{v,u}$;
  for $(u,v) \in E$ **do**
    $gap \leftarrow \alpha_{u,v}^{(t)} - \alpha_{v,u}^{(t)}$;
    if $gap \geq -w_{u,v}$ *and* $gap \leq w_{u,v}$ **then**
      $x_{u,v} \leftarrow \frac{w_{u,v}+gap}{2}$; $y_{v,u} \leftarrow \frac{w_{v,u}-gap}{2}$;
    else if $gap > w_{u,v}$ **then** $x_{u,v} \leftarrow w_{u,v}$; $y_{v,u} \leftarrow 0$;
    else $x_{u,v} \leftarrow 0$; $y_{v,u} \leftarrow w_{u,v}$;
  for $(u,v) \in E$ **do**
    $\alpha_{u,v}^{(t)} \leftarrow x_{u,v} + \frac{t-1}{t+2} \cdot (x_{u,v} - \alpha_{u,v}^{(t-1)})$;
    $\alpha_{v,u}^{(t)} \leftarrow y_{v,u} + \frac{t-1}{t+2} \cdot (y_{v,u} - \alpha_{v,u}^{(t-1)})$;
  foreach $u \in V$ **do** $r^{(t)}(u) \leftarrow \sum_{v \in N(u,G)} \alpha_{u,v}^{(t)}$;
// Extract the ADS
sort all $v \in V$ by weight $r^{(T)}(v)$;
$s^* \leftarrow \arg\max_{1 \leq i \leq n} \rho_R(V_i)$, where V_i is the top-i vertices in V;
return $G[V_{s^*}]$;

To address **Challenge 2**, we explicitly derive the gradient projection operator w.r.t. the function h, which governs the relationship between $\alpha_{u,v}$ and $\alpha_{v,u}$ as: $\texttt{prox}_h(\alpha_{u,v}, \alpha_{v,u}) =$

$$\begin{cases} \left(\frac{\alpha_{u,v} - \alpha_{v,u} + w_{u,v}}{2}, \quad \frac{\alpha_{v,u} - \alpha_{u,v} + w_{u,v}}{2}\right) & \text{if } |\alpha_{u,v} - \alpha_{v,u}| \le w_{u,v} \\ (w_{u,v}, 0) & \text{if } \alpha_{u,v} > \alpha_{v,u} + w_{u,v} \\ (0, w_{u,v}) & \text{if } \alpha_{u,v} < \alpha_{v,u} - w_{u,v} \end{cases}$$

Now, we are ready to present the PGD-based ADS algorithm, named PGD-ADS, which aims to optimize the $\text{QP}'(G, A)$ formulation via the PGD solver.

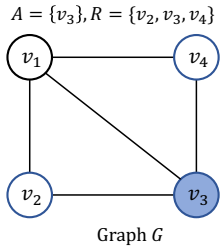
Algorithm 2 outlines the procedure for PGD-ADS. Initially, the algorithm sets the values of $\alpha^{(0)}_{u,v} = \frac{w_{u,v}}{2}$ for all edges $(u, v) \in E$ to satisfy the constraints of $\text{QP}'(G, A)$ and initializes $r^{(0)}(u)$ for every vertex $u \in V$ accordingly (see lines 1–2). The first two lines aim to initialize all variables in the $\text{QP}'(G, A)$ formulation. It subsequently sets the learning rate η (see theorem 4.3), which will be used throughout the iterative process (line 3). The main body of the algorithm consists of a for-loop spanning T iterations (lines 3–17), during which vertex weights are iteratively refined. Within each iteration, the algorithm first establishes the descent direction for the α variables (lines 5–7), providing the basis for the projected gradient step. Specifically, for every edge (u, v) in the t-th iteration, the algorithm calculates the difference between $\alpha^{(t)}_{u,v}$ and $\alpha^{(t)}_{v,u}$. Depending on the result, three distinct cases are analyzed, each corresponding to the proximal gradient of $(\alpha_{u,v}, \alpha_{v,u})$ with respect to the function h. These are then used to update auxiliary variables $x_{u,v}$ and $y_{v,u}$ (lines 9–14). Afterward, each $\alpha^{(t)}_{u,v}$ is updated for all edges (u, v), utilizing the new auxiliary variables together with their previous values from iteration $t - 1$ (lines 15–16). Next, the algorithm refreshes the values $r^{(t)}(u)$ for every vertex $u \in V$ (line 17). Finally, the ADS is extracted following steps similar to those in Algorithm 1 (lines 18-20).

Example 2. *Fig. 2 illustrates the vertex weight update process in* PASTA *with* $\eta = \frac{1}{12}$ *in a single iteration. Initially, edge weights are set to* $\alpha^{(0)}_{u,v} = \alpha^{(0)}_{v,u} = 0.5$ *for* $i = 1$ *to* 3*, and* $\alpha^{(0)}_{u,v} = \alpha^{(0)}_{v,u} = 1$ *for* $i = 4$ *to* 5*. Based on these edge weights, we compute the initial vertex weights. For example,* $r^{(0)}(v_1) = \alpha^{(0)}_{v_1,v_2} + \alpha^{(0)}_{v_1,v_3} + \alpha^{(0)}_{v_1,v_4} = \frac{3}{2}$*. In the first iteration, considering edge* e_1*, we compute:* $\alpha^{(1)}_{v_1,v_2} = \frac{1}{2}$*,* $\alpha^{(1)}_{v_2,v_1} = \frac{3}{8}$*, and* $gap = \frac{1}{8}$*. Then we update:* $x_{v_1,v_2} = \frac{1+gap}{2} = \frac{9}{16}$*, and* $y_{v_1,v_2} = \frac{1-gap}{2} = \frac{7}{16}$*. Next, we get:* $\alpha^{(1)}_{v_1,v_2} = x_{v_1,v_2} + 0 \cdot (x_{v_1,v_2} - \alpha^{(0)}_{v_1,v_2}) = \frac{9}{16}$*, and* $\alpha^{(1)}_{v_2,v_1} = x_{v_2,v_1} + \frac{1-1}{1+2} \cdot (x_{v_2,v_1} - \alpha^{(0)}_{v_2,v_1}) = \frac{7}{16}$*. After updating all* α *values, we update the vertex scores. For example,* $r^{(1)}(v_3) = \frac{7}{24} + \frac{41}{48} + \frac{41}{48} = 2$*. Finally, the* PAVA *extracts the subgraph induced by the vertex set* $S = \{v_1, v_2, v_3, v_4\}$*, which is returned as the ADS.*

We note that our PGD-ADS algorithm is formulated in the standard framework of PGD methods, specifically following the structure of the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA), which we introduce in detail in our supplementary material. Building on this foundation, we can establish the convergence rate of PGD-ADS from a theoretical perspective. Let us first review the theorem from [28], which provides the convergence rate of the FISTA algorithm for unconstrained optimization problems.

Theorem 4.2 ([28]). *Let* x^* *be the optimal solution of the following unconstrained optimization problem:* $\min_x f(x)$*, where* $f(x)$ *is a convex function with Lipschitz continuous gradient, and let* L_f *be the Lipschitz constant of* $\nabla f(x)$*. Supposing the learning rate* $\eta \le \frac{1}{L_f}$*, then after* T *iterations, the* FISTA *algorithm guarantees the following convergence rate:* $f(x^{(T)}) - f(x^*) \le \frac{2 \cdot \|x^{(0)} - x^*\|^2}{\eta \cdot T^2}$.

To leverage the above theorem for $\text{QP}'(G, A)$, we proceed as follows: (1) compute the Lipschitz constant of the gradient of $f(\alpha)$; (2) let α and α^* denote the current and optimal solutions of $f(\alpha)$



	$\alpha^{(0)}_{u,v}$	$\alpha^{(0)}_{v,u}$	$x_{u,v}$	$y_{v,u}$	$\alpha^{(1)}_{u,v}$	$\alpha^{(1)}_{v,u}$
(v_1, v_2)	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{9}{16}$	$\frac{7}{16}$	$\frac{9}{16}$	$\frac{7}{16}$
(v_1, v_3)	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{17}{24}$	$\frac{7}{24}$	$\frac{17}{24}$	$\frac{7}{24}$
(v_1, v_4)	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{9}{16}$	$\frac{7}{16}$	$\frac{9}{16}$	$\frac{7}{16}$
(v_2, v_3)	1	1	$\frac{55}{48}$	$\frac{41}{48}$	$\frac{55}{48}$	$\frac{41}{48}$
(v_3, v_4)	1	1	$\frac{41}{48}$	$\frac{55}{48}$	$\frac{41}{48}$	$\frac{55}{48}$

	$r^{(0)}(\cdot)$	$r^{(1)}(\cdot)$
v_1	$\frac{3}{2}$	$\frac{11}{6}$
v_2	$\frac{3}{2}$	$\frac{19}{2}$
v_3	$\frac{5}{2}$	2
v_4	$\frac{3}{2}$	$\frac{19}{2}$

Fig. 2. An Illustrative example of PASTA.

under the constraints of $\mathrm{QP}'(G, A)$; (3) use the bound on $f(\alpha) - f(\alpha^*)$ to derive the convergence rate of $\mathrm{QP}'(G, A)$.

Theorem 4.3. *The Lipschitz constant L_f of $\nabla f(\alpha)$, is bounded by $2 \cdot (|A| + 1) \cdot \sqrt{\Delta(\mathrm{G}) \cdot \mathrm{vol}(\mathrm{R})}$.*

Proof sketch. Based on the definition of Lipschitz constant, we have

$$L_f^2 = \|\nabla f(\alpha)\|^2 = \sum_{(u,v)\in E} \sum_{(x,y)\in E} \left(\frac{\partial \left(\nabla f(\alpha)_{u,v}\right)}{\partial \alpha_{x,y}} \right)^2.$$

And because,

$$\frac{\partial \left(\nabla f(\alpha)_{u,v}\right)}{\partial \alpha_{x,y}} = \frac{\partial \left(2 \cdot (\mathbb{1}[u \notin A] + g_{u,v}) \cdot r(u)\right)}{\partial \alpha_{x,y}} \leq \begin{cases} 2 \cdot (|A| + 1), & u = x \\ 0, & u \neq x \end{cases}$$

Therefore,

$$L_f^2 \leq \sum_{u\in V} \sum_{(u,v)\in E} \sum_{(u,y)\in E} (2 \cdot (|A| + 1))^2 \leq \sum_{u\in V} \sum_{(u,v)\in E} 4(|A| + 1)^2 \cdot d(u)$$
$$\leq 4(|A| + 1)^2 \Delta(G) \sum_{u\in V} \sum_{(u,v)\in E} 1 \leq 4(|A| + 1)^2 \Delta(G) \cdot \mathrm{vol}(R)$$

Therefore, $L_f \leq 2(|A| + 1)\sqrt{\Delta(G) \cdot \mathrm{vol}(R)}$. □

Now, we present the formal convergence rate of PGD-ADS. Notably, our analysis begins by introducing a new vector $\boldsymbol{\lambda} \in \mathbb{R}^n$, where for each $u \in V$, the corresponding entry in it is:

$$\lambda(u) = \begin{cases} r(u) & u \notin A \\ \max_{v\in V} r(v) & u \in A \end{cases}$$

That is, the objective function of $\mathrm{QP}'(G, A)$ is: $\min(\|\boldsymbol{\lambda}\|_2)^2$, based on which, we present the following theorem:

Theorem 4.4. *Given a graph $G = (V, E)$, and two vertex sets $R, A \subseteq V$. In* PGD-ADS*, when $T > O\left(\frac{\sqrt{|A|} \cdot \mathrm{vol}(R)^{\frac{1}{4}} \cdot \Delta(G)^{\frac{3}{4}} \cdot \sqrt{(\mathrm{vol}(R) + |A|\Delta(G))}}{\epsilon}\right)$, the following convergence guarantee holds: $\|\boldsymbol{\lambda}\|_\infty - \rho_R^* \leq \epsilon$.*

Preview. Note that Algorithm 2 is guaranteed to find a $(1 + \epsilon)$-approximation solution after $O\left(\frac{\sqrt{|A|} \cdot \mathrm{vol}(R)^{\frac{1}{4}} \cdot \Delta(G)^{\frac{3}{4}} \cdot \sqrt{(\mathrm{vol}(R) + |A|\Delta(G))}}{\epsilon}\right)$ iterations. To ensure that, the theorem 4.2 claims that the learning rate must satisfy $\eta \leq \frac{1}{L_f}$. For simplicity, and following the common practice in first-order

convex optimization [9], we directly set $\eta = \frac{1}{L_f}$ in line 3 of Algorithm 2. In addition, our algorithm is not highly sensitive to the learning rate η.

Proof sketch. We first provide our proof roadmap: (1) we prove that if $f(\boldsymbol{\alpha}) - f(\boldsymbol{\alpha}^*) \leq \mu$, and we have $\|\boldsymbol{\lambda}\|_\infty - \|\boldsymbol{\lambda}^*\|_\infty \leq \sqrt{\mu}$, and (2) let $\mu := \frac{2\cdot\|x^{(0)}-x^*\|^2}{\eta\cdot T^2} = \epsilon^2$. Then, by Theorem 4.2, we establish the relationship between the target accuracy ϵ and the number of iterations T, thus completing the proof. □

5 Our graph reduction technique

In this Section, we propose a simple yet effective graph reduction technique, which first reduces the search space into a much smaller graph.

5.1 A k-$r$core-based graph reduction technique

In this subsection, we propose a graph reduction technique to localize the ADS into a relatively small sub-graph before running an ADS discovery algorithm.

We note that while some existing works have proposed graph reduction techniques for the global DSD problem [81], such techniques remain surprisingly unexplored for the ADS problem. Moreover, the reduction methods developed for global DSD are not directly applicable to ADS, as the latter is defined with respect to two seed vertex sets, A and R, making its problem structure fundamentally different from that of global DSD. What we seek is a graph reduction technique that harmonizes the density requirement with the constraints of sets R and A, which can significantly reduce the search space for ADS discovery algorithms. In the following, we show how to achieve this by proposing a novel cohesive subgraph model k-$r$core.

Definition 5.1 (R-degree). Given a subgraph $G(H)$, and a reference vertex set $R \subseteq V$. The R-degree of $u \in H$ is defined as:

$$d_R(u, G[H]) = \begin{cases} |N(u, G[H \cap R])| + |N(u, G[H])| & u \in R \\ |N(u, G[H \cap R])| & u \notin R \end{cases}$$

Based on the R-degree, we define a novel cohesive subgraph model, called k-$r$core, which serves as the foundation for our graph reduction technique.

Definition 5.2 (k-$r$core). Given a graph $G = (V, E)$ and a reference vertex set R, its k-$r$core ($k > 0$), denoted by $\mathcal{M}$, is the largest subgraph of G such that $\forall u \in V(\mathcal{M}),\ d_R(u, \mathcal{M}) \geq k$.

Example 3. *Let us continue with the graph in Fig. 1, where $A = \{v_5\}$ and $R = \{v_5, v_7, v_8\}$. We also consider the subgraph S induced by the vertex set $\{v_4, v_5, v_6, v_7, v_8\}$. Thus, the R-degrees of all vertices in S are: $d_R(v_4, S) = d_R(v_6, S)$=3, and $d_R(v_5, S) = d_R(v_7, S) = d_R(v_8, S)$=6. That is, the subgraph S is a 3-$r$core.*

Theorem 5.3. *Given a graph $G = (V, E)$, two vertex sets R and A, and the $\lceil\rho_R^*\rceil$-$r$core, $\mathcal{M}^*$, the ADS $\mathcal{D} = G[S^*]$ must be contained within $G[V(\mathcal{M}^*) \cup A]$, where ρ_R^* is the NR-density of the ADS $\mathcal{D}$.*

Proof sketch. We prove this by contradiction. Suppose that the ADS $\mathcal{D}$=$G[S^*]$ is not fully contained in $G[V(\mathcal{H}^*) \cup A]$. Then there exists a vertex $u \in S^* \wedge u \notin A$ such that:

$$d_R(u, G[S^*]) < \rho_R^*.$$

Consider removing a vertex$u$ from S^*, and let $S' = S^* \setminus \{u\}$. Since $\mathcal{D}$ is an ADS, it holds that $\rho_R(G[S']) \leq \rho_R^*$. On the other hand, we can compute $\rho_R(G[S'])$ as:

$$\rho_R(G[S']) = \frac{2|E(S')|}{|S'|} = \frac{2|E(S^*)| - 2d_R(u, G[S^*])}{|S^*| - 1},$$

and since $d_R(u, G[S^*])$ counts the number of R-neighbors of u in $G[S^*]$, we further have:

$$2|E(S')| = \rho_R^* \cdot |S^*| - d_R(u, G[S^*]).$$

Therefore,

$$\rho_R(G[S']) = \frac{\rho_R^* \cdot |S^*| - d_R(u, G[S^*])}{|S^*| - 1}.$$

Since $\rho_R(G[S']) \leq \rho_R^*$, we obtain:

$$\frac{\rho_R^* \cdot |S^*| - d_R(u, G[S^*])}{|S^*| - 1} \leq \rho_R^* \Rightarrow d_R(u, G[S^*]) \geq \rho_R^*.$$

Thus, the theorem holds.

□

Thanks to the nested relationship of k-$r$core, i.e., for any two non-negative integers i and j with $i > j$, the i-$r$core is a subgraph of the j-$r$core. Thus, based on the above theorem, we derive the following corollary.

Corollary 5.4. *Given a graph $G = (V, E)$, and a $\lceil \underline{\rho_R} \rceil$-$r$core $\underline{\mathcal{M}}$, the ADS of G, $\mathcal{D}$ is contained in the subgraph $G[V(\underline{\mathcal{M}}) \cup A]$, where $\underline{\rho_R}$ is the lower bound of the $\rho_R(\mathcal{D})$ (i.e., ρ_R^*).*

Example 4. *Fig. 1 demonstrates the effectiveness of our graph reduction technique. Suppose $A = \{v_5\}$ and $R = \{v_5, v_7, v_8\}$, and $\underline{\rho_R}$=2.4. That is, the ADS must be contained in the $\lceil 2.4 \rceil$-$r$core, here, the 3-$r$core, denoted as $\underline{\mathcal{M}}$, is the subgraph induced by the vertex set $\{v_4, v_5, v_6, v_7, v_8\}$. By theorem 5.4, the ADS is guaranteed to be contained within the subgraph induced by the vertex set $\{v_4, v_5, v_6, v_7, v_8\}$, which is exactly the ADS of G.*

5.2 Efficiently estimating the lower bound of ρ_R^*

Recall that given a lower bound $\underline{\rho_R}$ of ρ_R^*, we can localize the graph within a small subgraph. Intuitively, the tighter the lower bound $\underline{\rho_R}$ we find, the smaller the resulting graph, thereby enabling a more efficient ADS discovery. In this subsection, we show how to estimate a decent lower bound of the optimal NR-density.

Lemma 5.5. *Given a graph G=(V, E), and a subgraph induced by the vertex set S, we have: $\sum_{v \in S} d_R(v, G[S]) = 2 \times (2|E(S)| - \sum_{v \in S \setminus R} d(v, G[S]))$.*

The detailed proof is shown in our supplementary material. Building upon the above lemma, we present a new result to establish a lower bound for ρ_R^* and the k-$r$core.

Lemma 5.6. *Given a graph $G = (V, E)$ and two vertex sets $R, A \subseteq V$, if there exists a k-$r$core $\mathcal{M}$ in G, then the NR-density of the ADS in G is at least $\frac{k^2}{2(k+|A|)}$.*

Proof sketch. We construct a subgraph $G[V(\mathcal{M}) \cup A]$, and derive a lower bound on its NR-density based on the minimum degree property of k-rcore. By analyzing the monotonicity of the resulting density expression, we obtain the bound $\frac{k^2}{2(k+|A|)}$. □

Building on Lemma 5.6, we can derive a lower bound of ρ_R^* as a function of the parameter k. To maximize the effectiveness of graph reduction, we aim to identify the largest value of k such that a k-$r$core exists, and use this value to obtain a tight lower bound on ρ_R^*. We present the details in Algorithm 3.

Specifically, it first computes the R-degree for each vertex $u \in V$ (line 1). Next, it iteratively peels the vertex with the minimum R-degree and updates its neighbors' R-degree (lines 3-9). Once a

Algorithm 3: LowerBound

Input : A graph $G = (V, E)$ and two vertex sets R, A
Output: The lower bound of the NR-density of the graph

1 **foreach** $u \in V$ **do** compute its R-degree $d_R(u, G)$;
2 $\underline{\rho_R} \leftarrow 0$;
3 **while** V *is not empty* **do**
4 $u \leftarrow \arg\min_{u \in V} d_R(u, G)$;
// update the k value
5 $k \leftarrow \min\{d_R(u, G), |V(G)|\}$;
6 $\underline{\rho_R} \leftarrow \max\{\underline{\rho_R}, \frac{k^2}{2 \cdot (k+|A|)}\}$; // set the lower bound of ρ_R^*
7 **foreach** $v \in N(u, G)$ **do**
8 $d_R(v, G) \leftarrow d_R(v, G) - w_{u,v}$;
9 remove u from G;
10 **return** $\underline{\rho_R}$;

vertex is removed from the graph, we update the lower bound of ρ_R^* via line 6. After all vertices are processed in the same way as described above, we return the maximum lower bound found during the entire process (line 10).

6 Our PASTA and PASTA-Exact algorithms

In this section, we present our PASTA and PASTA-Exact algorithms, both of which are based on the PGD-ADS algorithm and the $r$core-based graph reduction.

Algorithm 4: PASTA and PASTA-Exact

input : A graph $G = (V, E)$, two vertex sets R, A, a real value ϵ, and an integer T
output: An approximate/exact ADS $\mathcal{D}$
// compute the lower bound of ρ_R^*
1 $\underline{\rho_R} \leftarrow$ LowerBound(G, R, A);
2 $f \leftarrow$False, $\mathcal{D} \leftarrow \emptyset$;
3 **repeat**
// obtain a small graph
4 $\underline{\mathcal{M}} \leftarrow$ compute the $\lceil \underline{\rho_R} \rceil$-$r$core; $G' \leftarrow G[V(\underline{\mathcal{M}}) \cup A]$;
5 $\mathcal{D} \leftarrow$PGD-ADS(G', R, A) (see Algorithm 2);
// $\epsilon = 0$: PASTA-Exact; $\epsilon > 0$: PASTA
6 **if** $\mathcal{D}$ *satisfies* $(1+\epsilon)$*-approx. ratio* **then** $f \leftarrow$ True;
7 $\underline{\rho_R} \leftarrow \max\{\underline{\rho_R}, \rho_R(\mathcal{D})\}$; // update the lower bound of ρ_R^*
8 **until** f=True;
9 **return** $\mathcal{D}$;

Algorithm 4 gives an overview of PASTA (and PASTA-Exact when ϵ=0). They find the ADS by iteratively refining the search space and applying a PGD-based QP solver, PGD-ADS. They begin by estimating a lower bound $\underline{\rho_R}$ of the optimal anchored density ρ_R^* via LowerBound (line 1). Then, a while loop is used to find an ADS (lines 3-8). First, based on the $\underline{\rho_R}$, it first computes the $\lceil \underline{\rho_R} \rceil$-$r$core and based on which to reduce the search space (line 4). On this pruned graph, the core PGD-ADS subroutine is applied to generate a candidate ADS (line 5). If this candidate meets the $(1+\epsilon)$-approximation criterion, it is returned as the solution. Otherwise, the algorithm updates the lower bound and repeats the process (lines 3-8).

We note that our method leverages lower bounds of the optimal density ρ_R^* in two stages: 1) Before running the algorithm, we estimate an initial lower bound of ρ_R^* and apply the k-$r$core reduction accordingly. 2) During the execution of PASTA, our method adopts a progressive paradigm in which the current solution density ρ_R continuously improves over time. These updates progressively reduce the scale of the graph in subsequent iterations. In other words, even on datasets where the initial lower bound is relatively loose, the progressive updates ensure that graph reduction remains highly effective in later stages. In latter experiments, we observe that the reduction is substantial on all datasets. Moreover, the use of PGD-ADS as the primary solver brings notable theoretical improvements over the Frank-Wolfe-ADS, leading to significant gains in both convergence and overall performance.

Here, we provide the convergence rate of PASTA and compare it with the SOTA method FDP in terms of theoretical convergence rates:

Theorem 6.1. *Given a graph $G = (V, E)$, and two vertex sets $R, A \subseteq V$. In* PASTA, *when $T > O\left(\frac{\sqrt{|A|}\cdot\Delta(G)\cdot\sqrt{(\mathrm{vol}(R)+|A|\Delta(G))}}{\epsilon}\right)$, the algorithm is guaranteed to return a $(1+\epsilon)$-approximation solution.*

The result in Theorem 6.1 is straightforward, since PASTA is built upon PGD-ADS, and thus inherits its convergence properties. Specifically, the convergence rate of our method is significantly better than that of FDP, as shown in the following theorem:

Theorem 6.2. *To find a $(1+\epsilon)$-approximation solution,* PASTA *reduces the overall time complexity by at least $O\left(\frac{\mathrm{vol}(R)^{\frac{3}{4}}\cdot\Delta(G)^{\frac{5}{4}}}{\epsilon}\right)$ over* FDP, *where all variables have the same meanings as Theorem 4.4*

Conclusion: Our algorithm PASTA reduces the number of iterations by at least $O\left(\frac{\mathrm{vol}(R)^{\frac{3}{4}}\cdot\Delta(G)^{\frac{5}{4}}}{\epsilon}\right)$ compared to FDP, when a $(1+\epsilon)$-approximate solution is required.

7 Experiments

We now present the experimental results. Section 7.1 discusses the setup. We discuss the results in Sections 7.2 and 7.3.

Table 4. Datasets used in our experiments.

Dataset	Category	$\lvert V\rvert$	$\lvert E\rvert$
OpenFlights (OF)	Infrastructure	2,939	15,677
Ego-Facebook (EF)	Social	4,039	88,234
Email (EM)	Communication	265,009	364,481
Com-Amazon (CA)	Comments	334,863	925,872
DBLP (DP)	Co-author	317,080	1,049,866
Web-Google (WG)	Web	875,713	5,105,039
Youtube (YT)	Friendship	3,223,585	9,375,374
Cit-Patents (CP)	Citation	3,774,768	16,518,948
LiveJournal (LJ)	Social	4,847,571	68,993,773
UK-2002 (UK)	Web	18,520,486	298,113,762
IT-2004 (IT)	Hyperlink	41,291,594	1,027,474,947
SK-2005 (SK)	Web	50,636,154	1,949,412,601

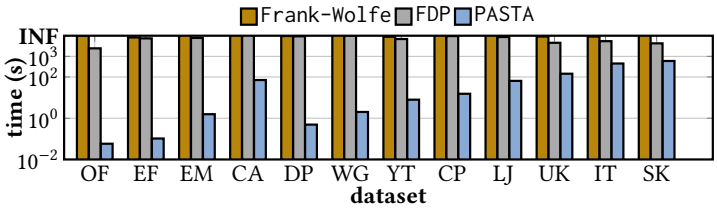


Fig. 3. Efficiency of approximation algorithms ($\epsilon = 0.1$).

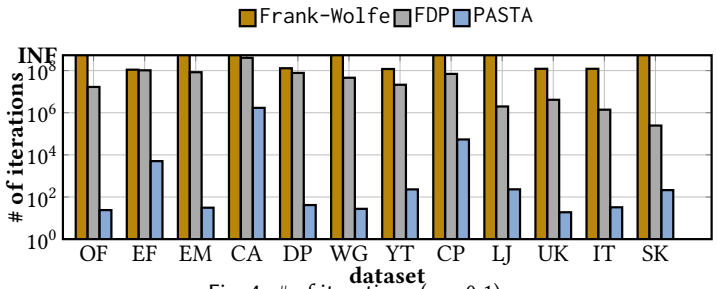


Fig. 4. # of iterations ($\epsilon = 0.1$).

7.1 Setup

7.1.1 ***Datasets.*** We use 12 real-world datasets from different domains, which are downloaded from the Stanford Network Analysis Platform [58], Laboratory of Web Algorithmics [55], Network Repository [60], and Konect [41]. Their detailed descriptions can also be found on these websites. Table 4 reports the statistics of these graphs.

7.1.2 ***Competitors.*** We evaluate the following algorithms for the ADS problem:

- `Frank-Wolfe` [37]: we adapt the basic `Frank-Wolfe` algorithm to solve the ADS problem, which is introduced in our supplementary material.
- `FDP` [74]: the state-of-the-art approximation algorithm, which is briefly recapped in Section 1.
- `FDPE` [74]: the state-of-the-art exact algorithm, which is discussed in Section 3.
- `PASTA`: our proposed approximation algorithm in Section 6.
- `PASTA-Exact`: our proposed exact algorithm in Section 6.

7.1.3 ***Settings.*** We implement all the algorithms in C++ and run experiments on a machine with an Intel(R) Xeon(R) Gold 6338 CPU @ 2.00GHz and 512GB of memory, with Ubuntu installed. If an algorithm cannot finish in 72 hours, we mark its running time as **INF** in the Figure and "—" in the Table. In our experiments, we follow the existing works [49, 62, 81]: For any ϵ, each $(1+\epsilon)$-approximation algorithm (including exact algorithms) runs for T iterations. After every T iterations, we assess whether the estimated error is below ϵ. For exact algorithms, we additionally check whether the optimality condition is satisfied. If the criterion is met, the algorithm terminates; otherwise, another T iterations are performed, repeating this process until convergence. In our comparison, we follow the existing works [49, 74] to set T=100.

7.1.4 ***Generation of*** A ***and*** R. To create the seed sets A and R, we employ the same strategy introduced in prior work [20, 74, 76]. This process commences by randomly selecting a vertex u from the vertex set V. The set A is then constituted by randomly choosing nodes from the immediate and secondary neighbors of u typically fixing the size of A at 8 (i.e., $|A|$=8). The set R is constructed through several random walks from A. The number and length of walks are determined by drawing random integers within the range of [3, 10]. For small networks (i.e., $m < 5 \times 10^8$), we generate 200

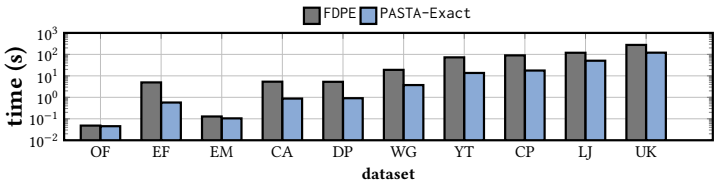


Fig. 5. Efficiency of exact algorithms.

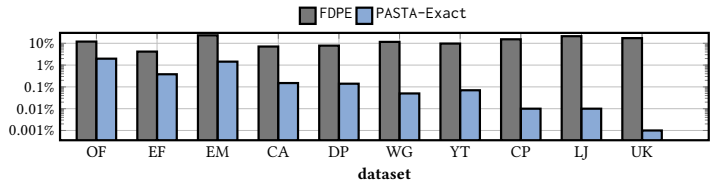


Fig. 6. Time cost of maximum flow in exact algorithms.

queries, while for large networks, we generate 100 queries to ensure efficiency. Unless otherwise specified, all reported results are averaged over these queries.

7.2 Efficiency results

In this section, we extensively compare PASTA and PASTA-Exact with the competitor algorithms from various angles.

• **1. Efficiency of approximation algorithms.** We evaluate the efficiency of three approximations on all datasets with ϵ=0.1. As shown in Fig. 3, we can see that: (1) Our algorithm PASTA is up to four orders of magnitude faster than the baseline methods, since it has a better theoretical guarantee and can find the ADS over a very small graph. For example, on the OF dataset, PASTA only takes 0.06s to find the 1.1-approximation solution, while FDP takes more than 2,400s (2) On more than half of the datasets, neither Frank-Wolfe nor FDP can find solutions that meet the accuracy requirement within three days, which is consistent with our theoretical analysis.

• **2. The number of iterations.** In this experiment, we report the number of iterations that each algorithm needed to find the near-optimal solution (i.e., $\epsilon = 0.1$) in Fig. 4. We make the following observations: (1) PASTA requires significantly fewer iterations to obtain approximate solutions compared to its competitors. For example, on the DP dataset, Frank-Wolfe, FDP, and PASTA require around 129M, 77M, and 41 iterations, respectively. (2) Compared to Frank-Wolfe, FDP requires fewer iterations. Although both algorithms are based on the standard Frank-Wolfe framework, FDP achieves faster convergence by enabling more balanced weight assignments. This phenomenon is also observed in other convex programming-based DSD solutions. [80, 81].

• **3. Efficiency of exact algorithms.** Fig. 5 presents the running time of all exact algorithms on the ten datasets. To be specific, PASTA-Exact performs best for all datasets, which is up to 10× faster than the competitors. This is because it significantly reduces the time cost incurred for computing maximum flow, and it can compute the minimum cut on the smaller flow network.

• **4. Time cost of the maximum flow in exact algorithms.** Recall that all of the exact ADS algorithms use maximum flow for optimal verification. Fig. 6 shows the time cost of the maximum flow step on ten datasets, and the graphs are assumed to be loaded into memory. We observe that PASTA-Exact achieves lower runtime for maximum flow computations, as it invokes the maximum flow algorithm less frequently and each invocation is restricted to a significantly smaller subgraph.

7.3 Effectiveness results

We extensively analyze the effectiveness of PASTA and PASTA-Exact from different angles.

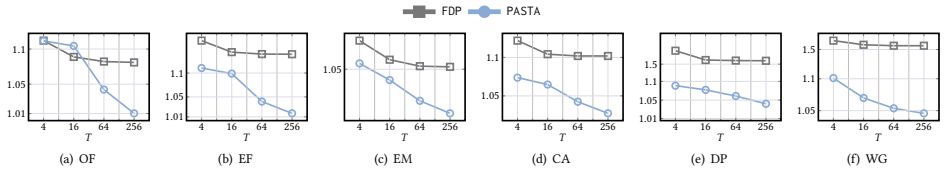


Fig. 7. Comparison of actual approximation ratios.

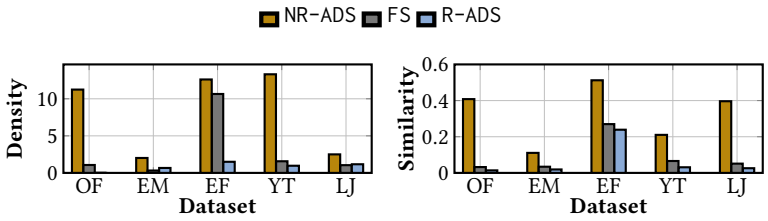


Fig. 8. Comparison of community quality.

Table 5. Comparison of BGCN and BGCN-ADS.

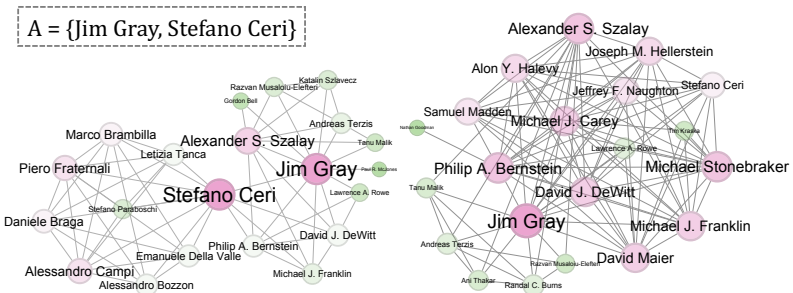
Dataset	Method	Recall@10	Recall@20	Recall@30	Recall@40
Netease	BGCN	0.0369	0.0642	0.0845	0.1013
	BGCN-ADS	**0.0404**	**0.0655**	**0.0865**	**0.1050**
Youshu	BGCN	0.1596	0.2410	0.2984	0.3416
	BGCN-ADS	**0.1811**	**0.2638**	**0.3216**	**0.3633**

Dataset	Method	NDCG@10	NDCG@20	NDCG@30	NDCG@40
Netease	BGCN	0.0202	0.0274	0.0321	0.0356
	BGCN-ADS	**0.0221**	**0.0288**	**0.0336**	**0.0374**
Youshu	BGCN	0.0934	0.1165	0.1303	0.1398
	BGCN-ADS	**0.1049**	**0.1281**	**0.1422**	**0.1515**

• **1. Actual approximation ratio.** We compute the actual approximation ratios of the approximation solutions returned by each algorithm on six datasets, where the number of iterations varies from 4 to 256. The ground-truth ADS densities are computed using our proposed exact ADS discovery algorithm, PASTA-Exact. We observe that our method PASTA outperforms FDP in terms of accuracy. For example, on the EF dataset, when T=256, our algorithm PASTA achieves a solution with a 1.01 approximation ratio, whereas the baseline algorithm FDP only reaches a 1.3-approximation ratio.

• **2. Community search.** As for community search, we compare ADS under NR-density (NR-ADS) with the state-of-the-art local community search method [70] (FS), as well as ADS under R-density (R-ADS). As shown in Fig. 8, NR-ADS consistently achieves higher community quality, reflected by higher density and stronger internal cohesion, measured by the Jaccard similarity between pairs of community members.

• **3. Bundle recommendation.** ADS can be used to enhance the performance of the state-of-the-art bundle product recommendation method BGCN [13], which aims to recommend a bundle of products to a single user. Specifically, BGCN [13] trains a Graph Neural Network (GNN)-based recommendation model using three networks, including the user-bundle interaction network, user-item interaction network, and bundle-item affiliation network, and then predicts whether a



(a) ADS under R-density (R-ADS) (b) ADS under NR-density (NR-ADS)

Fig. 9. A case study on DBLP co-authorship network.

user will interact with a certain bundle. However, due to exposure bias and item diversity imbalance [13], the bundle-item affiliation network cannot capture the precise relationships between bundles and items, which may cause some items to be overlooked and not considered for the bundle, leading to sparse connections between bundles and items.

To alleviate the sparsity issue above, we propose to detect ADSs from the item co-purchasing network and then use these ADSs to augment the bundle–item affiliation network, thereby enhancing the performance of BGCN. Specifically, we first build a co-purchasing network where an edge between two items indicates that they have been purchased together more than 5 times. Then, we apply PASTA (ϵ=0.1) to detect ADSs by treating the target items as the anchored set (A), and the items that frequently co-occur with them in the same bundles as the reference set (R). Afterwards, in the bundle–item affiliation network, we add links between each bundle and the items that belong to the same ADSs, enabling BGCN to leverage richer high-quality item associations. Finally, we run BGCN on the augmented networks, and denote this augmented approach by BGCN-ADS.

We have experimentally compared BGCN and BGCN-ADS on two real-world datasets, i.e., Netease and Youshu, whose statistics are shown in our supplementary material. We adopt two widely used metrics, i.e., Recall@K and NDCG@K, to judge the performance of the ranking list, where Recall@K measures the ratio of test bundles that have been contained by the top-K ranking list, while NDCG@K complements Recall by assigning higher scores to the hits at higher positions of the list. As shown in Table 5, BGCN-ADS outperforms BGCN in terms of Recall@K and NDCG@K. This is because it uses ADSs to augment the bundle-item affiliation network, which alleviates the sparsity issue by more precisely capturing the relationships between bundles and items.

• **4. Compared with other local density definitions.** We demonstrate that ADS obtained under NR-density (NR-ADS) is more effective than that based on R-density [20] (R-ADS) in downstream applications, such as bundle recommendation and community search. Especially, we observe that NR-ADS consistently outperforms R-ADS when used to augment the BGCN model, i.e., BGCN-NR-ADS achieves better performance than BGCN-R-ADS. Similarly, in community search, NR-ADS yields communities of higher quality than R-ADS. This advantage stems from the strong locality property of NR-density, which ensures that the discovered ADSs are more tightly centered around the anchored set. In contrast, R-density may include loosely related vertices, therefore degrading effectiveness. The detailed results are shown in our supplementary material due to space limitations.

• **5. Case studies.** We consider the DBLP co-authorship network and set the anchored set as A={*Jim Gray*, *Stefano Ceri*}. The reference set R is constructed by selecting the top-15 co-authors of these two researchers according to Google Scholar. We compare three communities: ADS under R-density (R-ADS), ADS under NR-density (NR-ADS), and the community returned by the SOTA local community search method (FS). As shown in Fig. 9, FS only returns the two anchored vertices

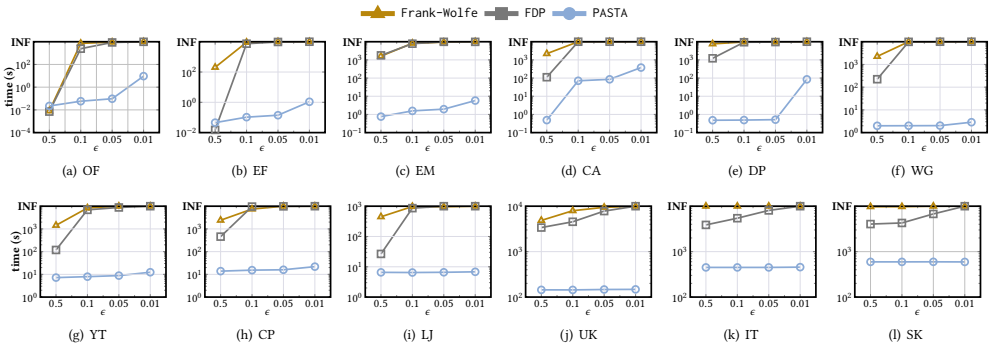


Fig. 10. Efficiency results of $(1+\epsilon)$-approximation algorithms.

in A, failing to expand a meaningful community. In contrast, NR-ADS discovers a significantly denser and more informative community than R-ADS. Moreover, NR-ADS includes several well-known database researchers, such as *Michael Stonebraker* and *Samuel Madden*, whereas R-ADS does not. This result indicates that NR-ADS better captures the core structure of the database research community, while remaining tightly centered around the anchored set. In addition, we also provide a case study on bundle recommendation, which is presented in the supplementary material.

7.4 Sensitivity analysis

In this subsection, we extensively analyze the sensitivity of our method PASTA.

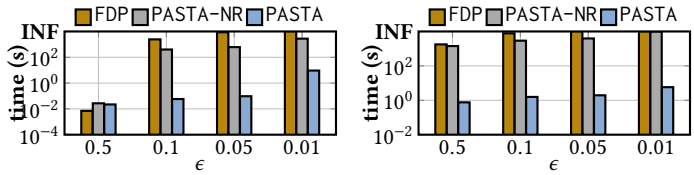
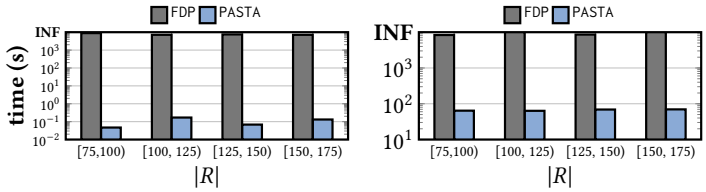


Fig. 11. Effect of graph reduction.

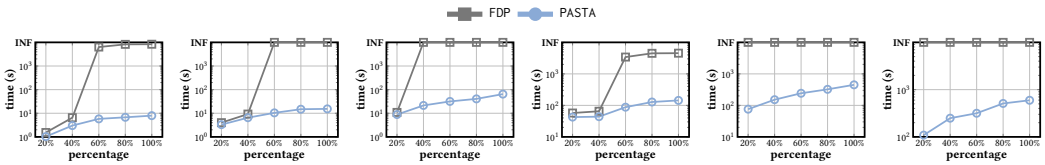
• **1. Effect of ϵ.** Fig. 10 compares the average running time of these three algorithms on 12 datasets by varying ϵ from 0.5 to 0.01. Clearly, PASTA is up to four orders of magnitude faster than Frank-Wolfe and FDP, since it has a better theoretical guarantee and can find ADSs on much smaller reduced graphs. Compared with Frank-Wolfe, FDP is up to two orders of magnitude faster when low accuracy is required (i.e., large ϵ). However, for high-accuracy settings, both methods often fail to complete within three days. In addition, even on very small graphs, PASTA is still more efficient than competitor methods because it consistently uses fewer iterations to achieve the same approximation ratio as FDP and Frank-Wolfe.

• **2. Effect of graph reduction technique.** We evaluate the effect of our graph reduction strategy on large datasets. We present the efficiency of PASTA (using k-$r$core-based graph reduction) and PASTA-NR (not using k-$r$core-based graph reduction). As shown in Fig. 11, we observe that PASTA is much faster than PASTA-NR, highlighting the effectiveness of our graph reduction technique in reducing the search space. We also note that even without the graph reduction technique, our PASTA-NR is still much faster than FDP.



(a) EF (b) LJ

Fig. 12. Effect of size R ($\epsilon = 0.1$).



(a) YT (b) CP (c) LJ (d) UK (e) IT (f) SK

Fig. 13. Scalability test for FDP and PASTA algorithms.

• **3. Effect of the size** R**.** We investigate the impact of the size of R on the performance of ADS algorithms, since the time complexity of all algorithms is positively correlated with $\text{vol}(R)$. As shown in Fig. 12, our algorithm remains stable as the size of R varies. Regardless of whether R is small or large, our algorithm consistently outperforms the baseline methods in terms of efficiency.

7.5 Memory usage and scalability

We compare the memory usage of different methods and analyze the scalability of PASTA.

• **1. Scalability test.** We evaluate the scalability of PASTA with $\epsilon = 0.1$ on the six largest datasets. Specifically, for each graph, we first randomly select 20%, 40%, 60%, 80%, and 100% of its vertices and then obtain six sub-graphs induced by these vertices, respectively. We then report the average efficiency of PASTA on these sub-graphs in Fig. 13. The time cost of PASTA scale is almost linear with the number of edges in the graph, since in each iteration, it takes $O(m)$ time.

• **2. Comparing the space usage of** FDP **and** PASTA**.** We examine the memory usage of FDP and PASTA on the five largest datasets, reporting their results in Table 6. FDP and PASTA have the same space complexity $O(m)$, where m denotes the number of edges in the graph. Nevertheless, in practice, PASTA usually takes less memory than FDP, since it adapts an effective graph reduction technique. This is an optimization technique at the practice level.

Table 6. Memory usage of FDP and PASTA.

Dataset	Memory Usage (KB)		Reduction ($\frac{\text{FDP}}{\text{PASTA}}$)
	FDP	PASTA	
WG	149,847,310	118,477,724	1.26×
YT	254,392,593	190,538,593	1.34×
CP	393,039,514	305,160,951	1.29×
LJ	1,185,994,782	873,890,514	1.36×
UK	4,865,509,000	2,798,672,000	1.73×

8 Related works

In this section, we review the related works, including the densest subgraph discovery (DSD) solutions and local community search.

• **Works for DSD problems.** The DSD problem aims to find the subgraph with the maximum average degree [3, 5, 6, 11, 12, 24, 32, 33, 45, 62, 66]. This problem can be addressed by solving a parametric maximum-flow [32]. While exact algorithms are often prohibitively time-consuming, especially on large graphs, many efforts have focused on developing approximation algorithms to improve efficiency [5, 14, 27, 40]. In particular, peeling-based approaches [12, 14, 15] and convex programming-based approximations [23, 33, 37, 54, 64] have been widely adopted for solving the DSD problem. Besides, the DSD problems have been extended to other types of graphs, including directed graphs [39, 50, 51, 62, 81, 83], multilayer graphs [30, 31, 38], bipartite graphs [1, 35, 52], uncertain graphs [85], hypergraphs graphs [4, 10], and heterogeneous graphs [17]. Many variants of DSD have also been studied [2, 20, 28, 34, 48, 59, 62, 64, 67, 72, 79, 80]. While these approaches focus on global graph search, recent work has increasingly emphasized local densest subgraph discovery. To enhance personalization in search results, the anchored densest subgraph (ADS) problem [20] aims to find the subgraph with the highest density while ensuring the presence of a user-defined anchor set. For the ADS problem, maximum-flow-based algorithms were first introduced [20, 76]. The convex programming-based algorithms have also been proposed [21, 74], which are extensively reviewed in Section 3. As a preview, we remark that their approaches significantly differ from ours.
• **Works for local community search.** Given a seed set or reference set R, the local community search problem aims to detect a community that is highly correlated to R [16, 25, 70, 73, 74]. Many works focus on identifying communities that achieve high scores according to specific quality metrics. Among these, two metrics are particularly prominent: conductance and density. Conductance-based metrics evaluate a community based on its cut and volume, where the "cut" denotes the number of outbound edges and the "volume" reflects the total degree of all community members [56, 70, 74]. In contrast, density-based metrics assess communities by the ratio of the "sum of weights"—influenced by the seed set R—to the size of the community [20, 21, 71, 74, 76]. The state-of-the-art metric for local community search, NR-density [21, 74], is the primary focus of our work. Besides, many cohesive subgraph models like k-core [8, 63, 65], k-truss [18, 61, 77], k-ECC [36, 75], k-clique [22], quasi-clique [68, 69, 78], and k-plex [7, 82] have been used in local community search [25].

9 Conclusions

In this paper, we investigate the problem of efficient anchored densest subgraph (ADS) discovery. Existing ADS solutions suffer from weaker theoretical guarantees and huge search spaces. To address these issues, we introduce a new approximation algorithm that requires fewer iterations to achieve the same solution accuracy as state-of-the-art (SOTA) methods, with a rigorous theoretical foundation. Next, we develop a novel graph reduction technique that effectively constrains ADS search to a smaller subgraph. Building on these advancements, we further propose an efficient exact algorithm. Experimental results on 12 real-world large directed graphs exhibit that our approach significantly outperforms existing methods, achieving up to four orders of magnitude speedup over SOTA algorithms. In future work, we plan to design efficient parallel ADS search algorithms and further extend our proposed methods to handle ADS discovery under alternative strongly local density measures. Incorporating ADS discovery into modern graph systems also presents an interesting future direction.

Acknowledgments

This work was supported by the 1+1+1 CUHK-CUHK(SZ)-GDSTC Joint Collaboration Fund under Grant 2025A0505000045.

References

[1] Reid Andersen. 2010. A local algorithm for finding dense subgraphs. *ACM TALG* 6, 4 (2010), 1–12.

[2] Reid Andersen and Kumar Chellapilla. 2009. Finding dense subgraphs with size bounds. In *WAW*. Springer, 25–37.

[3] Albert Angel, Nick Koudas, Nikos Sarkas, Divesh Srivastava, Michael Svendsen, and Srikanta Tirthapura. 2014. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *VLDB J.* 23, 2 (2014), 175–199.

[4] Naheed Anjum Arafat, Arijit Khan, Arpit Kumar Rai, and Bishwamittra Ghosh. 2023. Neighborhood-Based Hypergraph Core Decomposition. *Proceedings of the VLDB Endowment* 16, 9 (2023), 2061–2074.

[5] Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. 2012. Densest Subgraph in Streaming and MapReduce. *PVLDB* 5, 5 (2012).

[6] Oana Denisa Balalau, Francesco Bonchi, TH Hubert Chan, Francesco Gullo, and Mauro Sozio. 2015. Finding subgraphs with maximum total density and limited overlap. In *WSDM*. 379–388.

[7] Balabhaskar Balasundaram, Sergiy Butenko, and Illya V Hicks. 2011. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research* 59, 1 (2011), 133–142.

[8] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049* (2003).

[9] Amir Beck and Marc Teboulle. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences* 2, 1 (2009), 183–202.

[10] Suman K Bera, Sayan Bhattacharya, Jayesh Choudhari, and Prantar Ghosh. 2022. A new dynamic algorithm for densest subhypergraphs. In *Proceedings of the ACM Web Conference 2022*. 1093–1103.

[11] Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos Tsourakakis. 2015. Space-and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *STOC*. 173–182.

[12] Digvijay Boob, Yu Gao, Richard Peng, Saurabh Sawlani, Charalampos Tsourakakis, Di Wang, and Junxing Wang. 2020. Flowless: Extracting densest subgraphs without flow computations. In *WWW*.

[13] Jianxin Chang, Chen Gao, Xiangnan He, et al. 2020. Bundle recommendation with graph convolutional networks. In *Proceedings of the 43rd international ACM SIGIR conference on Research and development in Information Retrieval*. 1673–1676.

[14] Moses Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*. Springer, 84–95.

[15] Chandra Chekuri, Kent Quanrud, and Manuel R Torres. 2022. Densest Subgraph: Supermodularity, Iterative Peeling, and Flow. In *SODA*. SIAM, 1531–1555.

[16] Jiyang Chen, Osmar Zaïane, and Randy Goebel. 2009. Local community identification in social networks. In *2009 international conference on advances in social network analysis and mining*. IEEE, 237–242.

[17] Lu Chen, Chengfei Liu, Rui Zhou, Kewen Liao, Jiajie Xu, and Jianxin Li. 2023. Densest multipartite subgraph search in heterogeneous information networks. *Proceedings of the VLDB Endowment* 17, 4 (2023), 699–711.

[18] Jonathan Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. *National security agency technical report* 16, 3.1 (2008).

[19] Qiangqiang Dai, Rong-Hua Li, Hongchao Qin, Meihao Liao, and Guoren Wang. 2022. Scaling Up Maximal k-plex Enumeration. In *CIKM*. 345–354.

[20] Yizhou Dai, Miao Qiao, and Lijun Chang. 2022. Anchored Densest Subgraph. In *SIGMOD*. 1200–1213.

[21] Yizhou Dai, Miao Qiao, and Rong-Hua Li. 2024. On Density-based Local Community Search. *Proceedings of the ACM on Management of Data* 2, 2 (2024), 1–25.

[22] Maximilien Danisch, Oana Balalau, and Mauro Sozio. 2018. Listing k-cliques in sparse real-world graphs. In *WWW*. 589–598.

[23] Maximilien Danisch, T-H Hubert Chan, and Mauro Sozio. 2017. Large scale density-friendly graph decomposition via convex programming. In *WWW*. 233–242.

[24] Alessandro Epasto, Silvio Lattanzi, and Mauro Sozio. 2015. Efficient densest subgraph computation in evolving graphs. In *WWW*. 300–310.

[25] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2020. A survey of community search over big graphs. *VLDBJ* 29, 1 (2020), 353–392.

[26] Yixiang Fang, Wensheng Luo, and Chenhao Ma. 2022. Densest subgraph discovery on large graphs: Applications, challenges, and techniques. *Proceedings of the VLDB Endowment* 15, 12 (2022), 3766–3769.

[27] Yixiang Fang, Kaiqiang Yu, Reynold Cheng, Laks VS Lakshmanan, and Xuemin Lin. 2019. Efficient algorithms for densest subgraph discovery. *PVLDB* 12, 11 (2019), 1719–1732.

[28] Uriel Feige, Michael Seltser, et al. 1997. *On the densest k-subgraph problem.* Citeseer.

[29] Satoru Fujishige. 1980. Lexicographically optimal base of a polymatroid with respect to a weight vector. *Mathematics of Operations Research* 5, 2 (1980), 186–196.

[30] Edoardo Galimberti, Francesco Bonchi, and Francesco Gullo. 2017. Core decomposition and densest subgraph in multilayer networks. In *CIKM*. 1807–1816.

[31] Edoardo Galimberti, Francesco Bonchi, Francesco Gullo, and Tommaso Lanciano. 2020. Core decomposition in multilayer networks: theory, algorithms, and applications. *TKDD* 14, 1 (2020), 1–40.

[32] Andrew V Goldberg. 1984. *Finding a maximum density subgraph*. University of California Berkeley.

[33] Elfarouk Harb, Kent Quanrud, and Chandra Chekuri. 2022. Faster and Scalable Algorithms for Densest Subgraph and Decomposition. In *NIPS*.

[34] Yizhang He, Kai Wang, Wenjie Zhang, Xuemin Lin, and Ying Zhang. 2023. Scaling Up k-Clique Densest Subgraph Detection. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–26.

[35] Bryan Hooi, Hyun Ah Song, Alex Beutel, Neil Shah, Kijung Shin, and Christos Faloutsos. 2016. Fraudar: Bounding graph fraud in the face of camouflage. In *SIGKDD*. 895–904.

[36] Jiafeng Hu, Xiaowei Wu, Reynold Cheng, Siqiang Luo, and Yixiang Fang. 2016. Querying minimal steiner maximum-connected subgraphs in large graphs. In *CIKM*. 1241–1250.

[37] Martin Jaggi. 2013. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *ICML*. PMLR, 427–435.

[38] Vinay Jethava and Niko Beerenwinkel. 2015. Finding dense subgraphs in relational graphs. In *ECML PKDD*. Springer, 641–654.

[39] Ravindran Kannan and V Vinay. 1999. *Analyzing the structure of large graphs*. Forschungsinst. für Diskrete Mathematik.

[40] Samir Khuller and Barna Saha. 2009. On finding dense subgraphs. In *ICALP*. Springer, 597–608.

[41] Konect. 2006. Konect. http://konect.cc/networks/.

[42] Laks VS Lakshmanan. 2022. On a Quest for Combating Filter Bubbles and Misinformation. In *SIGMOD*. 2–2.

[43] Tommaso Lanciano, Atsushi Miyauchi, Adriano Fazzone, et al. 2024. A survey on the densest subgraph problem and its variants. *Comput. Surveys* 56, 8 (2024).

[44] Tommaso Lanciano, Atsushi Miyauchi, Adriano Fazzone, and Francesco Bonchi. 2023. A Survey on the Densest Subgraph Problem and its Variants. *arXiv preprint arXiv:2303.14467* (2023).

[45] Victor E Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. 2010. A survey of algorithms for dense subgraph discovery. In *Managing and mining graph data*. Springer, 303–336.

[46] Wensheng Luo, Chenhao Ma, Yixiang Fang, and Laks VS Lakshman. 2023. A Survey of Densest Subgraph Discovery on Large Graphs. *arXiv preprint arXiv:2306.07927* (2023).

[47] Wensheng Luo, Zhuo Tang, Yixiang Fang, Chenhao Ma, and Xu Zhou. 2023. Scalable Algorithms for Densest Subgraph Discovery. In *ICDE*. IEEE.

[48] Chenhao Ma, Reynold Cheng, Laks VS Lakshmanan, and Xiaolin Han. 2022. Finding locally densest subgraphs: a convex programming approach. *PVLDB* 15, 11 (2022), 2719–2732.

[49] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks VS Lakshmanan, and Xiaolin Han. 2022. A Convex-Programming Approach for Efficient Directed Densest Subgraph Discovery. In *SIGMOD*. 845–859.

[50] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks VS Lakshmanan, Wenjie Zhang, and Xuemin Lin. 2020. Efficient algorithms for densest subgraph discovery on large directed graphs. In *SIGMOD*. 1051–1066.

[51] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks VS Lakshmanan, Wenjie Zhang, and Xuemin Lin. 2021. On Directed Densest Subgraph Discovery. *TODS* 46, 4 (2021), 1–45.

[52] Michael Mitzenmacher, Jakub Pachocki, Richard Peng, Charalampos Tsourakakis, and Shen Chen Xu. 2015. Scalable large near-clique detection in large-scale networks via sampling. In *SIGKDD*. 815–824.

[53] Yu E Nesterov. 1983. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. In *Dokl. Akad. Nauk SSSR,*, Vol. 269. 543–547.

[54] Ta Duy Nguyen and Alina Ene. 2024. Multiplicative weights update, area convexity and random coordinate descent for densest subgraph problems. *arXiv preprint arXiv:2405.18809* (2024).

[55] Laboratory of Web Algorithmics. 2013. Laboratory of Web Algorithmics Datasets. http://law.di.unimi.it/datasets.php.

[56] Lorenzo Orecchia and Zeyuan Allen Zhu. 2014. Flow-based algorithms for local graph clustering. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 1267–1286.

[57] James B Orlin. 2013. Max flows in O (nm) time, or better. In *ACM Symposium on Theory of Computing*. 765–774.

[58] Stanford Network Analysis Project. 2009. SNAP. http://snap.stanford.edu/data/.

[59] Lu Qin, Rong-Hua Li, Lijun Chang, and Chengqi Zhang. 2015. Locally densest subgraph discovery. In *KDD*. 965–974.

[60] Network Repository. 2014. Network Repository. https://networkrepository.com/network-data.php.

[61] Kazumi Saito, Takeshi Yamada, and Kazuhiro Kazama. 2008. Extracting communities from complex networks by the k-dense method. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 91, 11 (2008), 3304–3311.

[62] Saurabh Sawlani and Junxing Wang. 2020. Near-optimal fully dynamic densest subgraph. In *STOC*. 181–193.

[63] Stephen B Seidman. 1983. Network structure and minimum degree. *Social networks* 5, 3 (1983), 269–287.

[64] Bintao Sun, Maximilien Danisch, TH Chan, and Mauro Sozio. 2020. KClist++: A Simple Algorithm for Finding k-Clique Densest Subgraphs in Large Graphs. *PVLDB* (2020).
[65] Longxu Sun, Xin Huang, Zheng Wu, and Jianliang Xu. 2024. Efficient Cross-layer Community Search in Large Multilayer Graphs. In *ICDE*. IEEE, 2959–2971.
[66] Nikolaj Tatti and Aristides Gionis. 2015. Density-friendly graph decomposition. In *WWW*. 1089–1099.
[67] Charalampos Tsourakakis. 2015. The k-clique densest subgraph problem. In *WWW*. 1122–1132.
[68] Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria Tsiarli. 2013. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *SIGKDD*. 104–112.
[69] Charalampos E Tsourakakis. 2014. Mathematical and algorithmic analysis of network and biological data. *arXiv preprint arXiv:1407.0375* (2014).
[70] Nate Veldt, Christine Klymko, and David F Gleich. 2019. Flow-based local graph clustering with better seed set inclusion. In *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 378–386.
[71] Yubao Wu, Ruoming Jin, Jing Li, and Xiang Zhang. 2015. Robust local community detection: on free rider effect and its elimination. *Proceedings of the VLDB Endowment* 8, 7 (2015), 798–809.
[72] Yichen Xu, Chenhao Ma, Yixiang Fang, and Zhifeng Bao. 2023. Efficient and Effective Algorithms for Generalized Densest Subgraph Discovery. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–27.
[73] Renchi Yang, Xiaokui Xiao, Zhewei Wei, Sourav S Bhowmick, Jun Zhao, and Rong-Hua Li. 2019. Efficient estimation of heat kernel pagerank for local clustering. In *Proceedings of the 2019 International Conference on Management of Data*. 1339–1356.
[74] Xiaowei Ye, Rong-Hua Li, Lei Liang, Zhizhen Liu, Longlong Lin, and Guoren Wang. 2024. Efficient and Effective Anchored Densest Subgraph Search: A Convex-programming based Approach. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3907–3918.
[75] Long Yuan, Lu Qin, Xuemin Lin, Lijun Chang, and Wenjie Zhang. 2017. I/O efficient ECC graph decomposition via graph reduction. *The VLDB Journal* 26, 2 (2017), 275–300.
[76] Qi Zhang, Yalong Zhang, Rong-Hua Li, and Guoren Wang. 2024. Approximate Anchored Densest Subgraph Search on Large Static and Dynamic Graphs. *Proceedings of the VLDB Endowment* 18, 3 (2024), 623–636.
[77] Yang Zhang and Srinivasan Parthasarathy. 2012. Extracting analyzing and visualizing triangle k-core motifs within networks. In *ICDE*. IEEE, 1049–1060.
[78] Yingli Zhou, Yixiang Fang, Chenhao Ma, Tianci Hou, and Xin Huang. 2024. Efficient Maximal Motif-Clique Enumeration over Large Heterogeneous Information Networks. *Proc. VLDB Endow.* 17, 11 (2024), 2946–2959.
[79] Yingli Zhou, Qingshuo Guo, and Yixiang Fang. 2025. Efficient k-Clique Densest Subgraph Discovery: Towards Bridging Practice and Theory. *Proceedings of the VLDB Endowment* 18, 10 (2025), 3490–3503.
[80] Yingli Zhou, Qingshuo Guo, Yixiang Fang, and Chenhao Ma. 2024. A Counting-based Approach for Efficient k-Clique Densest Subgraph Discovery. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–27.
[81] Yingli Zhou, Qingshuo Guo, Yi Yang, Yixiang Fang, Chenhao Ma, and Laks Lakshmanan. 2024. In-depth Analysis of Densest Subgraph Discovery in a Unified Framework. *arXiv preprint arXiv:2406.04738* (2024).
[82] Yi Zhou, Shan Hu, Mingyu Xiao, and Zhang-Hua Fu. 2021. Improving maximum k-plex solver via second-order reduction and graph color bounding. In *AAAI*, Vol. 35. 12453–12460.
[83] Yingli Zhou, Luocheng Liang, and Yixiang Fang. 2025. Efficient and Scalable Directed Densest Subgraph Discovery. *Proc. ACM Manag. Data* 3, 6 (2025), 1–27.
[84] Yingli Zhou, Youran Sun, and Yixiang Fang. 2025. Efficient Anchored Densest Subgraph Discovery: Improved Time Complexities and Practical Performance (technical report). https://github.com/JayLZhou/TechnicalReport/blob/main/ADS_Technical_report.pdf.
[85] Zhaonian Zou. 2013. Polynomial-time algorithm for finding densest subgraphs in uncertain graphs. In *MLG*.

Received October 2025; revised January 2026; accepted February 2026