Online Dense Subgraph Discovery via Blurred-Graph Feedback

Yuko Kuroki [1 2] **Atsushi Miyauchi** [1 2] **Junya Honda** [1 2] **Masashi Sugiyama** [2 1]

Abstract

Dense subgraph discovery aims to find a dense component in edge-weighted graphs. This is a fundamental graph-mining task with a variety of applications and thus has received much attention recently. Although most existing methods assume that each individual edge weight is easily obtained, such an assumption is not necessarily valid in practice. In this paper, we introduce a novel learning problem for dense subgraph discovery in which a learner queries edge subsets rather than only single edges and observes a noisy sum of edge weights in a queried subset. For this problem, we first propose a polynomial-time algorithm that obtains a nearly-optimal solution with high probability. Moreover, to deal with large-sized graphs, we design a more scalable algorithm with a theoretical guarantee. Computational experiments using real-world graphs demonstrate the effectiveness of our algorithms.

1. Introduction

Dense subgraph discovery aims to find a dense component in edge-weighted graphs. This is a fundamental graph-mining task with a variety of applications and thus has received much attention recently. Applications include detection of communities or span link farms in Web graphs (Dourisboure et al., 2007; Gibson et al., 2005), molecular complexes extraction in protein–protein interaction networks (Bader & Hogue, 2003), extracting experts in crowdsoucing systems (Kawase et al., 2019), and real-time story identification in micro-blogging streams (Angel et al., 2012).

Among a lot of optimization problems arising in dense subgraph discovery, the most popular one would be the *densest subgraph problem*. In this problem, given an edge-weighted undirected graph, we are asked to find a subset of vertices that maximizes the so-called *degree density* (or simply *density*), which is defined as half the average degree of the subgraph induced by the subset. Unlike most optimization problems for dense subgraph discovery, the densest subgraph problem can be solved exactly in polynomial time using some exact algorithms, e.g., Charikar's linear-programming-based (LP-based) algorithm (Charikar, 2000) and Goldberg's flow-based algorithm (Goldberg, 1984). Moreover, there is a simple greedy algorithm called the *greedy peeling*, which obtains a well-approximate solution in almost linear time (Charikar, 2000). Owing to the solvability and the usefulness of solutions, the densest subgraph problem has actively been studied in data mining, machine learning, and optimization communities (Ghaffari et al., 2019; Gionis & Tsourakakis, 2015; Miller et al., 2010; Papailiopoulos et al., 2014). We thoroughly review the literature in Appendix A.

Although the densest subgraph problem requires a full input of the graph data, in many real-world applications, the edge weights need to be estimated from *uncertain* measurements. For example, consider protein–protein interaction networks, where vertices correspond to proteins in a cell and edges (resp. edge weights) represent the interactions (resp. the strength of interactions) among the proteins. In the generation process of such networks, the edge weights are estimated through biological experiments using measuring instruments with some noises (Nepusz et al., 2012). As another example, consider social networks, where vertices correspond to users of some social networking service and edge weights represent the strength of communications (e.g., the number of messages exchanged) among them. In practice, we often need to estimate the edge weights by observing anonymized communications between users (Adar & Ré, 2007).

Recently, in order to handle the uncertainty of edge weights, Miyauchi & Takeda (2018) introduced a robust optimization variant of the densest subgraph problem. In their method, all edges are repeatedly queried by a sampling oracle that returns an individual edge weight. However, such a sampling procedure for individual edges is often quite costly or sometimes impossible. On the other hand, it is often affordable to observe aggregated information of a subset of edges. For example, in the case of protein–protein interaction networks, it may be costly to conduct experiments for all possible pairs of proteins, but it is cost-effective to observe molecular interaction among a molecular group

[1]The University of Tokyo, Japan [2]RIKEN AIP, Japan. Correspondence to: Yuko Kuroki <ykuroki@ms.k.u-tokyo.ac.jp>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

(Bader & Hogue, 2003). In the case of social networks, due to some privacy concerns and data usage agreements, it may be impossible even for data owners to obtain the estimated number of messages exchanged by two specific users, while it may be easy to access the information within some large group of users, because this procedure reveals much less information of individual users (Agrawal & Srikant, 2000; Zheleva & Getoor, 2011).

In this study, we introduce a novel learning problem for dense subgraph discovery, which we call *densest subgraph bandits* (*DS bandits*), by incorporating the concepts of *stochastic combinatorial bandits* (Chen et al., 2014; 2013) into the densest subgraph problem. In DS bandits, a learner is given an undirected graph, whose edge-weights are associated with unknown probability distributions. During the exploration period, the learner chooses a subset of edges (rather than only single edge) to sample, and observes the sum of noisy edge weights in a queried subset; we refer to this feedback model as *blurred-graph feedback*. We investigate DS bandits with the objective of *best arm identification*, that is, the learner must report one subgraph that she believes to be optimal after the exploration period.

Our learning problem can be seen as a novel variant of *combinatorial pure exploration* (CPE) problems (Chen et al., 2016; 2017; 2014). In the literature, most existing work on CPE has considered the case where the learner obtains feedback from each arm in a pulled subset of arms, i.e., the *semi-bandit* setting, or each individual arm can be queried (e.g. (Bubeck et al., 2013; Chen et al., 2017; 2014; Gabillon et al., 2012; Huang et al., 2018)). Thus, the above studies cannot deal with the aggregated reward from a subset of arms. On the other hand, existing work on the *full-bandit* setting has assumed that the objective function is linear and the size of subsets to query is exactly k at any round (Kuroki et al., 2020; Rejwan & Mansour, 2019), while our reward function (i.e., the degree density) is not linear and the size of subsets to query is not fixed in advance. If we fix the size of subsets to query to k in DS bandits, the corresponding offline problem (called the densest k-subgraph problem) becomes NP-hard and the best known approximation ratio is just $(1=n^{1=4+\ })$ for any > 0 (Bhaskara et al., 2010), where n is the number of vertices.

The contribution of this work is three-fold and can be summarized as follows.

1) We address a problem for dense subgraph discovery with no access to a sampling oracle for single edges (Problem 1) in the *fixed confidence setting*. For this problem, we present a general learning algorithm DS-Lin (Algorithm 2) based on the technique of *linear bandits* (Auer, 2003). We provide an upper bound of the number of samples that DS-Lin requires to identify an -optimal solution with probability at least 1 for > 0 and $2 (0; 1)$ (Theorem 1). Our key idea is to utilize an approximation algorithm (Algorithm 1) to compute the maximal confidence bound, thereby guaranteeing that the output by DS-Lin is an -optimal solution and the running time is polynomial in the size of a given graph.

2) To deal with large-sized graphs, we further investigate another problem with access to sampling oracle for any subset of edges (Problem 2) with a given fixed budget T. For this problem, we design a scalable and parameter-free algorithm DS-SR (Algorithm 3) that runs in $O(n^2T)$, while DS-Lin needs $O(m^2)$ time for updating the estimate, where m is the number of edges. Our key idea is to combine the *successive reject* strategy (Audibert et al., 2010) for the multi-armed bandits and the *greedy peeling* algorithm (Charikar, 2000) for the densest subgraph problem. We prove an upper bound on the probability that DS-SR outputs a solution whose degree density is less than $\frac{1}{2}$OPT , where OPT is the optimal value (Theorem 2).

3) In a series of experimental assessments, we thoroughly evaluate the performance of our proposed algorithms using well-known real-world graphs. We confirm that DS-Lin obtains a nearly-optimal solution even if the minimum size of queryable subsets is larger than the size of an optimal subset, which is consistent with the theoretical analysis. Moreover, we demonstrate that DS-SR finds nearly-optimal solutions even for large-sized instances, while significantly reducing the number of samples for single edges required by a state-of-the-art algorithm.

2. Problem Statement

In this section, we describe the densest subgraph problem and the online densest subgraph problem in the bandit setting formally.

2.1. Densest subgraph problem

The densest subgraph problem is defined as follows. Let $G = (V; E; w)$ be an undirected graph, consisting of $n = jVj$ vertices and $m = jEj$ edges, with an edge weight $w : E \to \mathbb{R}_{>0}$, where $\mathbb{R}_{>0}$ is the set of positive reals. For a subset of vertices $S \subseteq V$, let $G[S]$ denote the subgraph induced by S, i.e., $G[S] = (S; E(S))$ where $E(S) = \{\{u; v\} \in E : u; v \in S\}$. The *degree density* (or simply called the *density*) of $S \subseteq V$ is defined as $f_w(S) = w(S)=jSj$, where $w(S)$ is the sum of edge weights of $G[S]$, i.e., $w(S) = \sum_{e \in E(S)} w(e)$. In the densest subgraph problem, given an edge-weighted undirected graph $G = (V; E; w)$, we are asked to find $S \subseteq V$ that maximizes the density $f_w(S)$. There is an LP-based exact algorithm (Charikar, 2000), which is used in our proposed algorithm (see Appendix C for the entire procedure).

2.2. Densest subgraph bandits (DS bandits)

Here we formally define DS bandits. Suppose that we are given an (unweighted) undirected graph $G = (V, E)$. Assume that each edge $e \in E$ is associated with an unknown distribution ν_e over reals. $w : E \to \mathbb{R}_{>0}$ is the expected edge weights, where $w(e) = \mathbb{E}_{X \sim \nu_e}[X]$. Following the standard assumptions of stochastic multi-armed bandits, we assume that all edge-weight distributions have R-sub-Gaussian tails for some constant $R > 0$. Formally, if X is a random variable drawn from ν_e for $e \in E$, then for all $r \in \mathbb{R}$, X satisfies $\mathbb{E}[\exp(r(X - \mathbb{E}[X]))] \le \exp(R^2 r^2/2)$. We define the optimal solution as $S^* = \operatorname{argmax}_{S \subseteq V} f_w(S)$.

We first address the setting in which the learner can stop the game at any round if she can return an optimal solution for $\varepsilon > 0$ with high probability. Let $k > 2$ be the minimal size of queryable subsets of vertices; notice that the learner has no access to a sampling oracle for single edges. The problem is formally defined below.

Problem 1 (DS bandits with no access to single edges)**.** We are given an undirected graph $G = (V, E)$ and a family of queryable subsets of at least k (> 2) vertices $\mathcal{S} \subseteq 2^V$. Let $\varepsilon > 0$ be a required accuracy and $\delta \in (0, 1)$ be a confidence level. Then, the goal is to find $S_{\mathrm{OUT}} \subseteq V$ that satisfies $\Pr[f_w(S^*) - f_w(S_{\mathrm{OUT}}) \le \varepsilon] \ge 1 - \delta$, while minimizing the number of samples required by an algorithm (a.k.a. the sample complexity).

We next consider the setting in which the number of rounds in the exploration phase is fixed and is known to the learner, and the objective is to maximize the quality of the output solution. In this setting, we relax the condition of queryable subsets; assume that the learner is allowed to query any subset of edges. The problem is defined as follows.

Problem 2 (DS bandits with a fixed budget)**.** We are given an undirected graph $G = (V, E)$ and a fixed budget T. The goal is to find $S_{\mathrm{OUT}} \subseteq V$ that maximizes $f_w(S_{\mathrm{OUT}})$ within T rounds.

Note that Problem 1 is often called the fixed confidence setting and Problem 2 is called the fixed budget setting in the bandit literature.

3. Algorithm for Problem 1

In this section, we first present an algorithm for Problem 1 based on linear bandits, which we refer to as DS-Lin. We then show that DS-Lin is (ε, δ)-PAC, that is, the output of the algorithm satisfies $\Pr[f_w(S^*) - f_w(S_{\mathrm{OUT}}) \le \varepsilon] \ge 1 - \delta$. Finally, we provide an upper bound of the number of samples (i.e., the sample complexity).

3.1. DS-Lin algorithm

We first explain how to obtain the estimate of edge weights and confidence bounds. Then we discuss how to ensure a stopping condition and describe the entire procedure of DS-Lin.

Least-squares estimator. We construct an estimate of edge weight w using a sequential noisy observation. For $S \subseteq V$, let $\chi_{E(S)} \in \{0, 1\}^E$ be the indicator vector of $E(S) \subseteq E$, i.e., for each $e \in E$, $\chi_{E(S)}(e) = 1$ if $e \in E(S)$ and $\chi_{E(S)}(e) = 0$ otherwise. Therefore, each subset of edges $E(S)$ for $S \subseteq V$ corresponds to an arm whose feature is an indicator vector of it in linear bandits. For any $t > m$, we define a sequence of indicator vectors as $x_t = (\chi_{E(S_1)}, \ldots, \chi_{E(S_t)}) \in \{0, 1\}^{E \times t}$ and also define the corresponding sequence of observed rewards as $(r_1(S_1), \ldots, r_t(S_t)) \in \mathbb{R}^t$. We define A_{x_t} as

$$A_{x_t} = \sum_{i=1}^{t} \chi_{E(S_i)} \chi_{E(S_i)}^\top + \lambda I \in \mathbb{R}^{E \times E}$$

for a regularized term $\lambda > 0$, where I is the identity matrix. Let $b_{x_t} = \sum_{i=1}^{t} \chi_{E(S_i)} r_i(S_i) \in \mathbb{R}^E$. Then, the regularized least-squares estimator for $w \in \mathbb{R}^E$ can be obtained by

$$\hat{w}_t = A_{x_t}^{-1} b_{x_t} \in \mathbb{R}^E. \tag{1}$$

Confidence bounds. The basic idea to deal with uncertainty is that we maintain confidence bounds that contain the parameter $w \in \mathbb{R}^E$ with high probability. For a vector $x \in \mathbb{R}^m$ and a matrix $B \in \mathbb{R}^{m \times m}$, let $\|x\|_B = \sqrt{x^\top B x}$. Let $N(v) = \{u \in V : \{u, v\} \in E\}$ be the set of neighbors of $v \in V$ and $\deg_{\max} = \max_{v \in V} |N(v)|$ be the maximum degree of vertices. In the literature of linear bandits, Abbasi-Yadkori et al. (2011) proposed a high probability bound on confidence ellipsoids with a center at the estimate of unknown expected rewards. Plugging it into our setting, we have the following proposition on the ellipsoid confidence bounds for the estimate $\hat{w}_t = A_{x_t}^{-1} b_{x_t}$, where x_t is fixed beforehand:

Proposition 1 (Adapted from Abbasi-Yadkori et al. (2011), Theorem 2)**.** Let η_t be an R-sub-Gaussian noise for $R > 0$ and $R' = \sqrt{\deg_{\max}}\, R$. Let $\delta \in (0, 1)$ and assume that the ℓ_2-norm of edge weight w is less than L. Then, for any fixed sequence x_t, with probability at least $1 - \delta$, the inequality

$$|w(S) - \hat{w}_t(S)| \le C_t \|\chi_{E(S)}\|_{A_{x_t}^{-1}} \tag{2}$$

holds for all $t \in \{1, 2, \ldots\}$ and all $S \subseteq V$, where

$$C_t = R' \sqrt{2 \log \frac{\det(A_{x_t})^{\frac{1}{2}}}{\lambda^{\frac{m}{2}}}} + \lambda^{\frac{1}{2}} L. \tag{3}$$

The above bound can be used to guarantee the accuracy of the estimate.

Computing the maximal con dence bound. To identify a solution with an optimality guarantee, the learner ensures whether the estimate is valid by computing the maximal con dence bound among all subsets of vertices. We consider the following stopping condition:

$$f_{w_t}(S_t) \;\ge\; \max_{S \subseteq V :\, S \neq S_t} f_{w_t}(S) + \frac{C_t \max_{S \subseteq V} \|\mathbf{1}_{E(S)}\|_{A_{x_t}^{-1}}}{|S|} \quad : \tag{3}$$

The above stopping condition guarantees that the output satis es $f_w(S^*) - f_w(S_{\mathrm{OUT}}) \le \varepsilon$ with probability at least $1 - \delta$. However, computing $\max_{S \subseteq V} \|\mathbf{1}_{E(S)}\|_{A_{x_t}^{-1}}$ by brute force is intractable since it involves an exponential blow-up in the number of $S \subseteq V$. To overcome this computational challenge, we address a relaxed quadratic program:

$$\text{P1: } \max. \|x\|_{A_{x_t}^{-1}} \quad \text{s.t.} \quad \mathbf{e} \le x \le \mathbf{e}; \tag{4}$$

where $\mathbf{e} \in \mathbb{R}^m$ is the vector of all ones.

There is an ef cient way to solve P1 using the SDP-based algorithm by Ye (1999) for the following quadratic program with bound constraints:

$$\text{QP: } \max. \sum_{1 \le i,j \le m} q_{ij} x_i x_j \quad \text{s.t.} \quad \mathbf{e} \le x \le \mathbf{e}; \tag{5}$$

where $Q = (q_{ij}) \in \mathbb{R}^{m \times m}$ is a given symmetric matrix. Ye (1999) modi ed the algorithm by Goemans & Williamson (1995) and generalized the proof technique of Nesterov (1998), and then established the constant-factor approximation result for QP.

Proposition 2 (Ye (1999)) There exists a polynomial-time $\frac{4}{7}$-approximation algorithm for QP.

Note that Ye's algorithm (Ye, 1999) is a randomized algorithm, but it can be derandomized using the technique devised by Mahajan & Ramesh (1999). The learner can compute an upper bound of the maximal con dence bound $\max_{S \subseteq V} \|\mathbf{1}_{E(S)}\|_{A_{x_t}^{-1}}$ by using an approximate solution to QP obtained by the derandomized version of Ye's algorithm, because it is obvious that the optimal value of QP is larger than $\max_{S \subseteq V} \|\mathbf{1}_{E(S)}\|^2_{A_{x_t}^{-1}}$. Therefore, using Algorithm 1, we can ensure the following stopping condition in polynomial time:

$$f_{w_t}(S_t) \;\ge\; \max_{S \subseteq V :\, S \neq S_t} f_{w_t}(S) + \frac{C_t Z_t}{2} \quad ; \tag{6}$$

where Z_t denotes the objective value of the approximate solution to P1 and α is a constant-factor approximation ratio of Algorithm 1.

Algorithm 1 Unconstrained 0–1 quadratic programming

Input : A positive semide nite matrix $Q \in \mathbb{R}^{m \times m}$
Output : $x \in [-1, 1]^m$
Solve the following quadratic programming problem by Ye's algorithm (Ye, 1999) with derandomization (Mahajan & Ramesh, 1999):

$$\text{QP: } \max. \sum_{1 \le i,j \le m} q_{ij} x_i x_j \quad \text{s.t.} \quad \mathbf{e} \le x \le \mathbf{e};$$

and obtain a solution $x \in [-1, 1]^m$;
return x

Proposed algorithm. Let $T_t(S)$ be the number of times that $S \subseteq S$ is queried before t-th round in the algorithm. We present our algorithm DS-Lin, which is detailed in Algorithm 2. Our sampling strategy is based on a given allocation strategy p de ned as follows. Let $\mathcal{P}$ be a $|\mathcal{S}|$-dimensional probability simplex. We de ne p as $p = (p(S))_{S \in \mathcal{S}} \in \mathcal{P}$, where $p(S)$ describes the predetermined proportions of queries to a subset S. As a possible strategy, one can use the well-designed strategy called G-allocation (Pukelsheim, 2006; Soare et al., 2014), or simply use uniform allocation (see Appendix D for details). At each round t, the algorithm calls the sampling oracle for $S_t \in \mathcal{S}$ and observes $r_t(S_t)$. Then, the algorithm updates statistics A_{x_t} and b_{x_t}, and also updates the estimate w_t. To check the stopping condition, the algorithm approximately solves P1 by Algorithm 1 and computes the empirical best solution S_t using the LP-based exact algorithm for the densest subgraph problem for $G = (V, E; w_t)$. Once the stopping condition is satis ed, the algorithm returns the empirical best solution S_t as output.

3.2. Sample complexity

We prove that DS-Lin is (ε, δ)-PAC and analyze its sample complexity. We de ne the design matrix for $p \in \mathcal{P}$ as $_p = \sum_{S \in \mathcal{S}} p(S) \mathbf{1}_{E(S)} \mathbf{1}_{E(S)}^\top$. We de ne $_p$ as $_p = \max_{x \in [-1,1]^m} \|x\|^2_{(_p)^{-1}}$. Let $_{\min}$ be the minimal gap between the optimal value and the second optimal value, i.e., $_{\min} = \min_{S \subseteq V :\, S \neq S^*} f_w(S^*) - f_w(S)$. The next theorem shows an upper bound of the number of queries required by Algorithm 2 to output $S_{\mathrm{OUT}} \subseteq V$ that satis es $\Pr[f_w(S^*) - f_w(S_{\mathrm{OUT}}) \le \varepsilon] \ge 1 - \delta$.

Theorem 1. De ne $H = \frac{_p + \varepsilon}{(_{\min} + \varepsilon)^2}$. Then, with probability at least $1 - \delta$, DS-Lin (Algorithm 2) outputs $S \subseteq V$ whose density is at least $f_w(S^*) - \varepsilon$ and the total number of samples $$ is bounded as follows:

if $> 4m(\sqrt{m} + \sqrt{2})^2 \deg_{\max} R^2 H$, then

$$= O\left(\deg^2_{\max} R^2 \log \frac{1}{\delta} + L^2 \right) H \quad ;$$

Algorithm 2 DS-Lin

Input : Graph $G = (V, E)$, a family of queryable subsets of at least k (> 2) vertices $\mathcal{S} \subseteq 2^V$, parameter > 0, parameter $\in (0, 1)$, and allocation strategy p
Output : $S \subseteq V$
for $t = 1, \ldots, m$ **do**
  Choose $S_t \leftarrow \operatorname{argmin}_{S \in \mathrm{supp}(p)} \frac{T_t(S)}{p(S)}$;
  Call the sampling oracle for S_t;
  Observe $r_t(S_t)$;
  $b_{x_t} \leftarrow b_{x_t - 1} + \sum_{e \in E(S_t)} r_t(S_t)$;
end
while stopping condition (6) is not true **do**
  $t \leftarrow t + 1$;
  Choose $S_t \leftarrow \operatorname{argmin}_{S \in \mathrm{supp}(p)} \frac{T_t(S)}{p(S)}$;
  Call the sampling oracle for S_t and observe $r_t(S_t)$;
  $A_{x_t} \leftarrow A_{x_t - 1} + \sum_{e \in E(S_t)} \mathbf{x}_e \mathbf{x}_e^\top$;
  $b_{x_t} \leftarrow b_{x_t - 1} + \sum_{e \in E(S_t)} r_{S_t}$;
  $w_t \leftarrow A_{x_t}^{-1} b_t$;
  If $w_t(e) < 0$ then $w_t(e) = 0$ for each $e \in E$;
  $x \leftarrow$ Algorithm 1 for $A_{x_t}^{-1}$;
  $Z_t \leftarrow C_t \sqrt{\sum_{1 \le i,j \le m} A_{x_t}^{-1}(i,j) x_i x_j}$;
  $S_t \leftarrow$ Output of the LP-based exact algorithm (Charikar, 2000) for $G(V, E, w_t)$;
end
return $S_{\mathrm{OUT}} \leftarrow S_t$

and if $\frac{\deg_{\max} R^2}{L^2} \log \frac{1}{} $, then

$$= O\left(m \deg_{\max} R^2 H \log \frac{1}{} + C_{H,} \right),$$

where $C_{H,}$ is

$$O\left(m \deg_{\max} R^2 H \log \deg_{\max} R m H \log \frac{1}{} \right).$$

The proof of Theorem 2 is given in Appendix F. Note that $_p = d$ holds if we are allowed to query any subset of vertices and employ G-allocation strategy, i.e., $p = \operatorname{argmin}_{p \in P} \max_{S \subseteq V} k \mathbf{x}_{E(S)} k^2_{p^{-1}}$, which was shown in Kiefer & Wolfowitz (1960). However, in practice, we should restrict the size of the support to reduce the computational cost; finding a family of subsets of vertices that minimizes $_p$ may be also related to the optimal experimental design problem (Pukelsheim, 2006).

In the work of Chen et al. (2014), they proved that the lower bound on the sample complexity of general combinatorial pure exploration problems with linear rewards is $\left(\sum_{e \in [m]} \frac{1}{^2_e} \log \frac{1}{} \right)$, where m is the number of base arms and $_e$ is defined as follows. Let $\mathcal{M}$ be any decision class (such as size-k paths, matchings, and matroids). Let M^* be an optimal subset, i.e., $M^* = \operatorname{argmax}_{M \in \mathcal{M}} \sum_{e \in M} w_e$. For each base arm $e \in [m]$, the gap $_e$ is defined as $_e = \sum_{e \in M^*} w_e - \max_{M \in \mathcal{M} : e \notin M} \sum_{e \in M} w_e$ (if $e \notin M^*$), and $_e = \sum_{e \in M^*} w_e - \max_{M \in \mathcal{M} : e \in M} \sum_{e \in M} w_e$ (if $e \in M^*$).

In the work of Huang et al. (2018), they studied the combinatorial pure exploration problem with continuous and separable reward functions, and showed that the problem has a lower bound $\left(H \log \frac{1}{} + H m \log \log \frac{1}{} \right)$, where $H = \sum_{i=1}^{m} \frac{1}{^2_i}$. In their definition of H, the term $_i$ is called consistent optimality radius and it measures how far the estimate can be away from true parameter while the optimal solution in terms of the estimate is still consistent with the true optimal one in the i-th dimension (see Definition 2 in (Huang et al., 2018)).

Note that the problem settings in Chen et al. (2014) and Huang et al. (2018) are different from ours; in fact, in our setting the learner can query a subset of edges rather than a base arm and reward function is not linear. Therefore, their lower bound results are not directly applicable to our problem. However, we can see that our sample complexity in Theorem 1 is comparable with their lower bounds because ours is $O(H \log \frac{1}{} + H \log(H \log \frac{1}{}))$ if we ignore the terms irrespective of H and .

4. Algorithm for Problem 2

In this section, we propose a scalable and parameter-free algorithm for Problem 2 that runs in $O(n^2 T)$ time for a given budget T, and provide theoretical guarantees for the output of the algorithm.

4.1. DS-SR algorithm

The design of our algorithm is based on the Successive Reject (SR) algorithm, which was designed for a regular multi-armed bandits in the fixed budget setting (Audibert et al., 2010) and is known to be the optimal strategy (Carpentier & Locatelli, 2016). In classical SR algorithm, we divide the budget T into $K - 1$ (K is the number of arms) phases. During each phase, the algorithm uniformly samples an active arm that has not been dismissed yet. At the end of each phase, the algorithm dismisses the arm with the lowest empirical mean. After K phases, the algorithm outputs the last surviving arm.

For DS bandits, we employ a different strategy from the classical one because our aim is to find the best subset of vertices in a given graph. Specifically, our algorithm DS-SR is inspired by the graph algorithm called greedy peeling (Charikar, 2000), which was designed for approximately solving the densest subgraph problem. DS-SR removes one vertex in each phase, and after all phases are over, it selects the best subset of vertices according to the empirical observation.

Algorithm 3 DS-SR

Input : Budget $T > 0$, graph $G(V, E)$, sampling oracle
Output: $S \subseteq V$
$\widetilde{\log}(n) \leftarrow \sum_{i=1}^{n-1} \frac{1}{i}$;
$T_0 \leftarrow 0$;
For $T_0(v) \leftarrow 0$ for each $v \in V$;
$S_n \leftarrow V$ and $v_0 \leftarrow$;
for $t = 1, \ldots, n-1$ **do**
  $T_t \leftarrow \left\lfloor \frac{T}{\widetilde{\log}(n-1)(n-t)} \right\rfloor$;
  $T'_t \leftarrow \frac{T_t}{2|S_{n-t+1}|}$ and $\epsilon_t \leftarrow T'_t - T'_{t-1}$;
  for $v \in S_{n-t+1}$ **do**
    Run Algorithm 4 (sampling procedure);
  end
  $f(S_{n-t+1}) \leftarrow \frac{\frac{1}{2}\sum_{v \in S_{n-t+1}} \widehat{\deg}_{S_{n-t+1}}(v;t)}{|S_{n-t+1}|}$;
  $v_t \leftarrow \operatorname{argmin}_{v \in S_{n-t+1}} \widehat{\deg}_{S_{n-t+1}}(v;t)$;
  $S_{n-t} \leftarrow S_{n-t+1} \setminus \{v_t\}$;
end
return $S_{OUT} \in \{S_2, \ldots, S_n\}$ that maximizes $f(S_i)$

Algorithm 4 Sampling procedure (subroutine of Algorithm 3)

if $N_{S_{n-t+1}}(v) = \emptyset$ **then**
  Set $\widehat{\deg}_{S_{n-t+1}}(v;t) = 0$;
end
else
  if $v \in N_{S_{n-t+2}}(v_{t-1})$ **then**
    Sample $E_{S_{n-t+1}}(v)$ for ϵ_t times;
    $Y_t \leftarrow Y_{t-1} + \epsilon_t \widehat{X}_{E_{S_{n-t+1}}(v)}(t-1)$;
    $\widehat{\deg}_{S_{n-t+1}}(v;t) \leftarrow \frac{Y_t + \epsilon_t \widehat{X}_{E_{S_{n-t+1}}(v)}(t-1)}{T_{E_{S_{n-t+1}}(v)}(t-1) + \epsilon_t}$;
    $T_{E_{S_{n-t+1}}(v)}(t) \leftarrow T_{E_{S_{n-t+1}}(v)}(t-1) + \epsilon_t$;
  end
  else
    Sample $E_{S_{n-t+1}}(v)$ for $\sum_{i=1}^{t} \epsilon_i$ times;
    $\widehat{\deg}_{S_{n-t+1}}(v;t) \leftarrow \widehat{X}_{E_{S_{n-t+1}}(v)}(\sum_{i=1}^{t} \epsilon_i)$;
    $T_{E_{S_{n-t+1}}(v)}(t) \leftarrow \sum_{i=1}^{t} \epsilon_i$;
  end
end

Notation. For $S \subseteq V$ and $v \in S$, let $N_S(v) = \{u \in S : \{u, v\} \in E\}$ be the set of neighboring vertices of v in $G[S]$ and let $E_S(v) = \{\{u, v\} \in E : u \in N_S(v)\}$ be the set of incident edges to v in $G[S]$. For $F \subseteq 2^E$ and for all phases $t \geq 1$, we denote by $T_F(t)$ the number of times that F was sampled over all rounds from 1 to t, and denote by $X_F(1), \ldots, X_F(T_F(t))$ the sequence of associated observed weights. Introduce $\widehat{X}_F(k) = \frac{1}{k}\sum_{s=1}^{k} X_F(s)$ as the empirical mean of weights of F after k samples. For simplicity, we denote $\widehat{\deg}_{S,v}(t) = \widehat{X}_{E_S(v)}(T_{E(S)}(t))$.

Proposed algorithm. All procedures of DS-SR are detailed in Algorithm 3. Intuitively, DS-SR proceeds as follows. Given a budget T, we divide T into $n-1$ phases. DS-SR maintains a subset of vertices. Initially, $S_n = V$. In each phase t, for $v \in S_{n-t+1}$, the algorithm uses the sampling oracle for obtaining the estimate of the degree $\widehat{\deg}_{S_{n-t+1}}(v)$, which we refer to as the empirical degree. After the sampling procedure, we compute the empirical quality function $f(S_{n-t+1})$ and specify one vertex v_t that should be removed. In Algorithm 4, we detail the sampling procedure for obtaining the empirical degree $v \in S_{n-t+1}$. If v was not a neighbor of v_{t-1} in phase $t-1$, the algorithm samples $E_{S_{n-t+1}}(v)$ for ϵ_t times, where ϵ_t is set carefully. On the other hand, if v was a neighbor of that, the algorithm samples $E_{S_{n-t+1}}(v)$ for $\sum_{i=1}^{t} \epsilon_i$ times. Our eliminate scheme removes a vertex v_t that minimizes the empirical degree, i.e., $v_t \in \operatorname{argmin}_{v \in S_{n-t+1}} \widehat{\deg}_{S_{n-t+1}}(v;t)$. Finally, after $n-1$ phases have been done, DS-SR outputs $S_{OUT} \subseteq V$ that maximizes the empirical quality function, i.e., $S_{OUT} = \operatorname{argmax}_{S_i \in \{S_2, \ldots, S_n\}} f(S_i)$.

4.2. Upper bound on the probability of error

We provide an upper bound on the probability that the quality of solution obtained by the proposed algorithm is less than $\frac{1}{2} f_w(S^*)$, as shown in the following theorem.

Theorem 2. Given any $T > m$, and assume that the edge weight distribution ν_e for each arm $e \in [m]$ has mean $w(e)$ with an R-sub-Gaussian tail. Then, DS-SR (Algorithm 3) uses at most T samples and outputs $S_{OUT} \subseteq V$ such that

$$\Pr\left[f_w(S_{OUT}) < \frac{f_w(S^*)}{2}\right] \leq C_{G,\nu} \exp\left(-\frac{(T - \sum_{i=1}^{n+1} i)^2}{4n^2 \deg_{\max} R^2 \widetilde{\log}(n-1)}\right), \quad (7)$$

where $C_{G,\nu} = \frac{2\deg_{\max}(n+1)^3 2^n R^2}{\epsilon^2}$ and $\widetilde{\log}(n-1) = \sum_{i=1}^{n-1} \frac{1}{i}$.

The proof of Theorem 2 is given in Appendix H. From the theorem, we see that DS-SR requires a budget of $T = O\left(\frac{n^3 \deg_{\max}}{\epsilon^2} \log \frac{\deg_{\max}}{\epsilon}\right)$ by setting the RHS of (7) to a constant. Besides, the upper bound on the probability of error is exponentially decreasing with T.

5. Experiments

In this section, we examine the performance of our proposed algorithms DS-Lin and DS-SR. First, we conduct experiments for DS-Lin and show that DS-Lin can find a nearly-optimal solution without sampling any single edges.

Table 1. Real-world graphs used in our experiments.

Name	n	m	Description
Karate	34	78	Social network
Lesmis	77	254	Social network
Polbooks	105	441	Co-purchased network
Adjnoun	112	425	Word-adjacency network
Jazz	198	2,742	Social network
Email	1,133	5,451	Communication network
email-Eu-core	986	16,064	Communication network
Polblogs	1,222	16,714	Blog hyperlinks network
ego-Facebook	4,039	88,234	Social network
Wiki-Vote	7,066	100,736	Wikipedia "who-votes-whom"

Second, we perform experiments for DS-SR and demonstrate that DS-SR is applicable to large-sized graphs and significantly reduces the number of samples for single edges, compared to that of the state-of-the-art algorithm. Throughout our experiments, to solve the LPs in Charikar's algorithm (Charikar, 2000), we used a state-of-the-art mathematical programming solver, Gurobi Optimizer 7.5.1, with default parameter settings. All experiments were conducted on a Linux machine with 2.6 GHz CPU and 130 GB RAM. The code was written in Python.

Dataset. Table 1 lists real-world graphs on which our experiments were conducted. Most of those can be found on Mark Newman's website[1] or in SNAP datasets[2]. For each graph, we construct the edge weight w using the following simple rule, which is inspired by the knockout densest subgraph model introduced by Miyauchi & Takeda (2018). Let $G = (V, E)$ be an unweighted graph and let $S^* \subseteq V$ be an optimal solution to the densest subgraph problem. For each $e \in E$, we set $w(e) = \mathrm{rand}(1, 20)$ if $e \in E(S^*)$, and $w(e) = \mathrm{rand}(1, 100)$ if $e \in E \setminus E(S^*)$, where $\mathrm{rand}(\cdot, \cdot)$ is the function that returns a real value selected uniformly at random from the interval between the two values. That is, we set a relatively small value for each $e \in E(S^*)$ and a relatively large value for each $e \in E \setminus E(S^*)$, which often makes the densest subgraph on $G = (V, E)$ no longer densest on the edge-weighted graph $G = (V, E, w)$. Throughout our experiments, we generate a random noise $\zeta(e) \sim \mathcal{N}(0, 1)$ for all $e \in E$.

5.1. Experiments for DS-Lin

Baseline. We compare our algorithm with the following naive approach, which we refer to as Naive. As well as our proposed algorithm, Naive is a kind of algorithm that sequentially accesses a sampling oracle to estimate w and uses uniform sampling strategy. The entire procedure is detailed in Algorithm 5.

[1] http://www-personal.umich.edu/~mejn/netdata/

[2] http://snap.stanford.edu/

Algorithm 5 Baseline algorithm (Naive)

Input : Number of iterations T and a family of queryable subsets of at least k vertices $\mathcal{S} \subseteq 2^V$
Output : $S \subseteq V$

$w_{\mathrm{avg}} \leftarrow 0$;
$t_e \leftarrow 0$ for $e \in E$;
for $t = 1, 2, \ldots, T$ **do**
  Choose $S_t \in \mathcal{S}$ uniformly at random;
  Call the sampling oracle for S_t and observe $\zeta_t(S_t)$;
  $t_e \leftarrow t_e + 1$ for $e \in E(S_t)$;
  Update $w_{\mathrm{avg}}(e) \leftarrow \frac{w_{\mathrm{avg}}(e)(t_e - 1) + r_{S_t = \ell}}{t_e}$ for $e \in E(S_t)$;
end
$S \leftarrow$ Output of Charikar's LP-based exact algorithm (Charikar, 2000) for $G(V, E, w_{\mathrm{avg}})$;
return S

Table 2. Comparison between DS-Lin and the baseline algorithm (Algorithm 5).

Graph	k	DS-Lin	Naive	OPT	$\lvert S \rvert$
Karate	10	111.08	19.94	111.08	6
	20	111.08	19.94		
	30	111.08	19.94		
Lesmis	10	179.72	177.19	179.72	15
	20	179.72	177.19		
	30	179.72	177.19		
Polbooks	10	227.43	172.69	228.67	19
	20	227.62	172.69		
	30	227.67	172.69		
Adjnoun	10	133.23	53.27	134.83	55
	40	133.62	53.27		
	70	133.53	53.27		
Jazz	10	598.39	170.03	599.43	42
	40	598.81	170.46		
	70	598.81	164.76		
Email	10	223.36	67.24	223.90	58
	40	223.37	67.24		
	70	222.29	67.24		

Parameter settings. Here we use the graphs with up to ten thousand edges. We set the minimum size of queryable subsets $k = 10, 20, 30$ for Karate, Lesmis, and Polbooks, and $k = 10, 40, 70$ for Adjnoun, Jazz, and Email. We construct $\mathcal{S}$ so that the matrix consisting of rows corresponding to the indicator vectors of $S \in \mathcal{S}$ has rank m. Each $S \in \mathcal{S}$ is given as follows. We select an integer $\ell \in [k, n]$ and choose $S \subseteq V$ of size ℓ uniformly at random. A uniform allocation strategy is employed by DS-Lin as p, i.e., $p = (1/|\mathcal{S}|)_{S \in \mathcal{S}}$. We set $\alpha = 100$ and $R = 1$. In our theoretical analysis, we provided an upper bound of the number of queries required by DS-Lin for $\varepsilon > 0$ and $\delta \in (0, 1)$. However, such an upper bound is usually too large in practice. Therefore, we terminate the while-loop of our algorithm once the number of iterations exceeds 10,000

Table 3. Performance of DS-SR. For DS-SR and R-Oracle, the quality of solutions, number of samples, and computation time are averaged over 100 executions.

Graph	DS-SR				R-Oracle			G-Oracle	OPT
	T	Quality	#Samples for single edges	Time(s)	Quality	#Samples for single edges	Time(s)		
Karate	10^3	111.08	58	0.00	111.08	10,296	0.02	111.08	111.08
Lesmis	10^4	177.66	752	0.02	179.72	51,816	0.07	176.29	179.72
Polbooks	10^4	227.43	419	0.02	228.67	214,767	0.22	227.47	228.67
Adjnoun	10^4	133.93	403	0.02	134.83	241,400	0.26	133.97	134.83
Jazz	10^5	599.42	6,837	0.4	599.43	1,115,994	1.49	599.43	599.43
Email	10^6	220.7	23,785	1.51	223.91	22,790,631	20.54	220.93	223.90
email-Eu-core	10^6	792.03	34,393	4.0	792.19	17,509,760	29.69	792.07	792.19
Polblogs	10^6	1211.37	16,508	4.38	1211.44	18,452,256	20.76	1211.44	1211.44
ego-Facebook	10^7	2654.40	103,546	42.61	2783.85	78,175,324	108.82	2654.44	2783.85
Wiki-Vote	10^8	1235.71	3,975,994	425.42	1235.95	288,205,696	638.92	1235.76	1235.95

except for the initialization steps. To be consistent, we also set $T = m + 10000$ in Naive.

Results. Here we compare our proposed algorithm DS-Lin with Naive in terms of the quality of solutions. The results are summarized in Table 2. The quality of output S is measured by its density in terms of w, which is unknown to the learner. For all instances, we run each algorithm for 10 times, and report the average value. The last two columns of Table 2 represent the optimal value and the size of an optimal solution, respectively, As can be seen, our algorithm outperforms the baseline algorithm; in fact, our algorithm always obtains a nearly-optimal solution. It should be noted that this trend is valid even if k is quite large; in particular, even if k is larger than the size of the densest subgraph on the edge-weighted graph $G = (V, E, w)$, our algorithm succeeds in detecting a vertex subset that is almost densest in terms of w. We also report how the density of solutions approaches to such a quality and behavior of DS-Lin with respect to the number of iterations in Appendix I.

Finally, we brie y report the running time of our proposed algorithm with 10,000 iterations. For small-sized instances, Karate , Lesmis , Polbooks , and Adjnoun , the algorithm runs in a few minutes. For medium-sized instances, Jazz and Email , the algorithm runs in a few hours.

5.2. Experiments for DS-SR

Compared algorithms. To demonstrate the performance of DS-SR for Problem 2, we also implement two algorithms G-Oracle and R-Oracle. G-Oracle is the greedy peeling algorithm with the knowledge of the expected weight w (Charikar, 2000), which is detailed in Algorithm 6. Note that we are interested in how the quality of solutions by DS-SR is close to that of G-Oracle. R-Oracle is the state-of-the-art robust optimization algorithm proposed by Miyauchi & Takeda (2018) with the use of edge-weight space $W = \{w' \mid \forall e \in E,\ w'(e) \in [\min\{w(e) - 1, 0\}, w(e) + 1]\}$, which is detailed in Algorithm 7 in Appendix J. For R-Oracle, we set $\delta = 0.9$ and $\varepsilon = 0.9$ as in Miyauchi & Takeda (2018).

Algorithm 6 Greedy peeling (G-Oracle)

Input : Graph $G = (V, E, w)$
Output : $S \subseteq V$
$S_{|V|} \leftarrow V$;
for $i = |V|, \ldots, 2$ **do**
  Find $v_i \in \operatorname{argmin}_{v \in S_i} \deg_{S_i}(v)$;
  $S_{i-1} \leftarrow S_i \setminus \{v_i\}$;
end
return $S_i \in \{S_1, \ldots, S_{|V|}\}$ that maximizes $f_w(S)$

Results. For DS-SR, in order to make T_t positive, we run the experiments with a budget $T = 10^{\lceil \log_{10} \sum_{i=1}^{n+1} i \rceil}$ for all instances. The results are summarized in Table 3. The quality of output is again evaluated by its density in terms of w. For DS-SR and R-Oracle, we list the total number of samples for individual edges used in the algorithms. To observe the scalability, we also report the computation time of the algorithms. We perform them 100 times on each graph. As can be seen, DS-SR required much less samples for single edges than that of R-Oracle but still can find high-quality solutions. The quality of solutions by DS-SR is comparable with that of G-Oracle, which has a prior knowledge of expected weights w. Moreover, in terms of computation time, DS-SR efficiently works on large-sized graphs with about ten thousands of edges. Finally, Figure 1 depicts the fraction of the size of edge subsets queried in DS-SR (see Appendix J for results on all graphs). We see that in the execution of DS-SR, the fraction of the number of queries for single edges is less than 30%.

6. Conclusion

In this study, we introduced a novel online variant of the densest subgraph problem by bringing the concepts of com-

Figure 1. Fraction of the size of edge subsets queried by DS-SR. All values are averaged over 100 executions.

binatorial pure exploration, which we refer to as the DS bandits. We first proposed a (n, ε)-PAC algorithm called DS-Lin, and provided a polynomial sample complexity guarantee. Our key technique is to utilize an approximation algorithm using SDP for confidence bound maximization. Then, to deal with large-sized graphs, we proposed an algorithm called DS-SR by combining the successive reject strategy and the greedy peeling algorithm. We provided an upper bound of probability that the quality of the solution obtained by the algorithm is less than $\frac{1}{2}\mathrm{OPT}$. Computational experiments using well-known real-world graphs demonstrate the effectiveness of our proposed algorithm.

Acknowledgments

The authors thank the anonymous reviewers for their useful comments and suggestions to improve the paper. YK would like to thank Wei Chen and Tomomi Matsui for helpful discussion, and also thank Yasuo Tabei, Takeshi Teshima, and Taira Tsuchiya for their feedback on the manuscript. YK was supported by Microsoft Research Asia D-CORE program, KAKENHI 18J23034, and UTokyo Toyota-Dwango Scholarship. AM was supported by KAKENHI 19K20218. JH was supported by KAKENHI 18K17998. MS was supported by KAKENHI 17H00757.

References

Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. Improved algorithms for linear stochastic bandits. In *Proc. NIPS '14*, pp. 2312–2320, 2011.

Adar, E. and Re, C. Managing uncertainty in social networks. *IEEE Data Engineering Bulletin*, 30:15–22, 2007.

Agrawal, R. and Srikant, R. Privacy-preserving data mining. In *Proc. SIGMOD '00*, pp. 439–450, 2000.

Andersen, R. and Chellapilla, K. Finding dense subgraphs with size bounds. In *Proc. WAW '09*, pp. 25–37, 2009.

Angel, A., Sarkas, N., Koudas, N., and Srivastava, D. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. In *Proc. VLDB '12*, pp. 574–585, 2012.

Audibert, J.-Y., Bubeck, S., and Munos, R. Best arm identification in multi-armed bandits. In *Proc. COLT '10*, pp. 41–53, 2010.

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3:397–422, 2003.

Bader, G. D. and Hogue, C. W. V. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(1):1–27, 2003.

Bahmani, B., Kumar, R., and Vassilvitskii, S. Densest subgraph in streaming and mapreduce. In *Proc. VLDB '12*, pp. 454–465, 2012.

Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U., and Vijayaraghavan, A. Detecting high log-densities: An $O(n^{1/4})$ approximation for densest k-subgraph. In *Proc. STOC '10*, pp. 201–210, 2010.

Bhattacharya, S., Henzinger, M., Nanongkai, D., and Tsourakakis, C. E. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proc. STOC '15*, pp. 173–182, 2015.

Bouhtou, M., Gaubert, S., and Sagnol, G. Submodularity and randomized rounding techniques for optimal experimental design. *Electronic Notes in Discrete Mathematics*, 36:679–686, 2010.

Bubeck, S., Wang, T., and Viswanathan, N. Multiple identifications in multi-armed bandits. In *Proc. ICML '13*, pp. 258–265, 2013.

Carpentier, A. and Locatelli, A. Tight (lower) bounds for the fixed budget best arm identification bandit problem. In *Proc. COLT '16*, pp. 590–604, 2016.

Charikar, M. Greedy approximation algorithms for finding dense components in a graph. In *Proc. APPROX '00*, pp. 84–95, 2000.

Chen, L., Gupta, A., and Li, J. Pure exploration of multi-armed bandit under matroid constraints. In *Proc. COLT '16*, pp. 647–669, 2016.

Chen, L., Gupta, A., Li, J., Qiao, M., and Wang, R. Nearly optimal sampling algorithms for combinatorial pure exploration. In *Proc. COLT '17*, pp. 482–534, 2017.

Chen, S., Lin, T., King, I., Lyu, M. R., and Chen, W. Combinatorial pure exploration of multi-armed bandits. In *Proc. NIPS '14*, pp. 379–387, 2014.

Chen, W., Wang, Y., and Yuan, Y. Combinatorial multi-armed bandit: General framework and applications. In *Proc. ICML '13*, pp. 151–159, 2013.

Dourisboure, Y., Geraci, F., and Pellegrini, M. Extraction and classi cation of dense communities in the web. In *Proc. WWW '07*, pp. 461–470, 2007.

Epasto, A., Lattanzi, S., and Sozio, M. Ef cient densest subgraph computation in evolving graphs. In *Proc. WWW '15*, pp. 300–310, 2015.

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

Gabillon, V., Ghavamzadeh, M., and Lazaric, A. Best arm identi cation: A uni ed approach to xed budget and xed con dence. In *Proc. NIPS '12*, pp. 3212–3220, 2012.

Galimberti, E., Bonchi, F., and Gullo, F. Core decomposition and densest subgraph in multilayer networks. In *Proc. CIKM '17*, pp. 1807–1816, 2017.

Ghaffari, M., Lattanzi, S., and Mitrović, S. Improved parallel algorithms for density-based network clustering. In *Proc. ICML '19*, pp. 2201–2210, 2019.

Gibson, D., Kumar, R., and Tomkins, A. Discovering large dense subgraphs in massive graphs. In *Proc. VLDB '05*, pp. 721–732, 2005.

Gionis, A. and Tsourakakis, C. E. Dense subgraph discovery: KDD 2015 Tutorial. In *Proc. KDD '15*, pp. 2313–2314, 2015.

Goemans, M. X. and Williamson, D. P. Improved approximation algorithms for maximum cut and satis ability problems using semide nite programming. *Journal of the ACM*, 42:1115–1145, 1995.

Goldberg, A. V. Finding a maximum density subgraph. Technical report, University of California Berkeley, 1984.

Hu, S., Wu, X., and Chan, T.-H. H. Maintaining densest subsets ef ciently in evolving hypergraphs. In *Proc. CIKM '17*, pp. 929–938, 2017.

Huang, W., Ok, J., Li, L., and Chen, W. Combinatorial pure exploration with continuous and separable reward functions and its applications. In *Proc. IJCAI '18*, pp. 2291–2297, 2018.

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

Kawase, Y., Kuroki, Y., and Miyauchi, A. Graph mining meets crowdsourcing: Extracting experts for answer aggregation. In *Proc. IJCAI'19*, pp. 1272–1279, 2019.

Khuller, S. and Saha, B. On nding dense subgraphs. In *Proc. ICALP '09*, pp. 597–608, 2009. ISBN 978-3-642-02926-4.

Kiefer, J. and Wolfowitz, J. The equivalence of two extremum problems. *Canadian Journal of Mathematics*, 12: 363366, 1960.

Kuroki, Y., Xu, L., Miyauchi, A., Honda, J., and Sugiyama, M. Polynomial-time algorithms for multiple-arm identi cation with full-bandit feedback. *Neural Computation*, 32(9):1733–1773, 2020.

Mahajan, S. and Ramesh, H. Derandomizing approximation algorithms based on semide nite programming. *SIAM Journal on Computing*, 28:1641–1663, 1999.

McGregor, A., Tench, D., Vorotnikova, S., and Vu, H. T. Densest subgraph in dynamic graph streams. In *Proc. MFCS '15*, pp. 472–482, 2015.

Miller, B., Bliss, N., and Wolfe, P. Subgraph detection using eigenvector L1 norms. In *Proc. NIPS '10*, 2010.

Mitzenmacher, M., Pachocki, J., Peng, R., Tsourakakis, C. E., and Xu, S. C. Scalable large near-clique detection in large-scale networks via sampling. In *Proc. KDD '15*, pp. 815–824, 2015.

Miyauchi, A. and Kakimura, N. Finding a dense subgraph with sparse cut. In *Proc. CIKM '18*, pp. 547–556, 2018.

Miyauchi, A. and Takeda, A. Robust densest subgraph discovery. In *Proc. ICDM '18*, pp. 1188–1193, 2018.

Miyauchi, A., Iwamasa, Y., Fukunaga, T., and Kakimura, N. Threshold in uence model for allocating advertising budgets. In *Proc. ICML '15*, pp. 1395–1404, 2015.

Nasir, M. A. U., Gionis, A., Morales, G. D. F., and Girdzijauskas, S. Fully dynamic algorithm for top-k densest subgraphs. In *Proc. CIKM '17*, pp. 1817–1826, 2017.

Nepusz, T., Yu, H., and Paccanaro, A. Detecting overlapping protein complexes in protein-protein interaction networks. *Nature Methods*, 9(5):471–472, 2012.

Nesterov, Y. Semide nite relaxation and nonconvex quadratic optimization. *Optimization Methods and Software*, 9(1-3):141–160, 1998.

Papailiopoulos, D. S., Mitliagkas, I., Dimakis, A. G., and Caramanis, C. Finding dense subgraphs via low-rank bilinear optimization. In *Proc. ICML '14*, pp. 1890–1898, 2014.

Pukelsheim, F. *Optimal Design of Experiments*. SIAM, 2006.

Rejwan, I. and Mansour, Y. Top-k combinatorial bandits with full-bandit feedback. arXiv preprint arXiv:1905.12624, 2019.

Sagnol, G. Approximation of a maximum-submodular-coverage problem involving spectral functions, with application to experimental design. *Discrete Applied Mathematics*, 161:258–276, 2013.

Soare, M., Lazaric, A., and Munos, R. Best-arm identification in linear bandits. In *Proc. NIPS'14*, pp. 828–836, 2014.

Tsourakakis, C. E. The k-clique densest subgraph problem. In Proc. WWW '15, pp. 1122–1132, 2015.

Tsourakakis, C. E., Chen, T., Kakimura, N., and Pachocki, J. Novel dense subgraph discovery primitives: Risk aversion and exclusion queries. In *Proc. ECML-PKDD '19*, 2019. 17 pages.

Xu, L., Honda, J., and Sugiyama, M. A fully adaptive algorithm for pure exploration in linear bandits. In *Proc. AISTATS '18*, pp. 843–851, 2018.

Ye, Y. Approximating quadratic programming with bound constraints. *Mathematical Programming*, 84:219–226, 1999.

Zheleva, E. and Getoor, L. Privacy in social networks: A survey. In *Social Network Data Analytics*, pp. 277–306. Springer US, 2011.

Zou, Z. Polynomial-time algorithm for finding densest subgraphs in uncertain graphs. In *Proc. MLG '13*, 2013. 7 pages.

Supplementary Material:
Online Dense Subgraph Discovery via Blurred-Graph Feedback

A. Related work on the densest subgraph problem

The densest subgraph problem has a large number of noteworthy problem variations. The most related one is the above-mentioned variant dealing with the uncertainty of edge weights, recently introduced by Miyauchi & Takeda (2018). Here we review their models in detail. They introduced two optimization models: the robust densest subgraph problem and the robust densest subgraph problem with sampling oracle. In both models, it is assumed that we have an edge-weight space $\mathcal{W} = \prod_{e \in E} [l_e, r_e] \subseteq \prod_{e \in E} [0, 1]$ that contains the unknown true edge weight w. That is, we have only the lower and upper bounds on the true edge weight for each edge. The key question they addressed is as follows: how can we evaluate the quality of solutions in this uncertain scenario? To answer this question, they employed the measure called the robust ratio, which is a well-known notion in the field of robust optimization. In the first model, given an unweighted graph $G = (V, E)$ and an edge-weight space $\mathcal{W}$, we are asked to find $S \subseteq V$ that maximizes the robust ratio. In the second model, as mentioned above, we also have access to the edge-weight sampling oracle.

There are other problem variations under uncertain scenarios. Zou (2013) studied the densest subgraph problem on uncertain graphs. Uncertain graphs are a generalization of graphs, which can model the uncertainty of the existence of edges (rather than the uncertainty of edge weights). More formally, an uncertain graph consists of an unweighted graph $G = (V, E)$ and a function $p : E \to [0, 1]$, where $e \in E$ is present with probability $p(e)$ whereas $e \in E$ is absent with probability $1 - p(e)$. In the problem introduced by Zou (2013), given an uncertain graph, we are asked to find $S \subseteq V$ that maximizes the expected value of the density. Zou (2013) demonstrated that this problem can be reduced to the original densest subgraph problem, and designed polynomial-time exact algorithm using the reduction. Very recently, Tsourakakis et al. (2019) introduced a novel optimization model, which they refer to as the risk-averse DSD. In this model, given an uncertain graph, we are asked to find $S \subseteq V$ that has a large expected value of the density, at the same time, has a small risk. The risk of S is measured by the probability that S is not dense on a given uncertain graph. They showed that the risk-averse DSD can be reduced to NEG-DSD, and designed an efficient approximation algorithm based on the reduction.

In addition to the above uncertain variants, the densest subgraph problem has many other interesting variations. In particular, the size-restricted variants have been actively studied (Andersen & Chellapilla, 2009; Bhaskara et al., 2010; Feige et al., 2001; Khuller & Saha, 2009). For example, in the densest k-subgraph problem (Feige et al., 2001), given an edge-weighted graph G and a positive integer k, we are asked to find $S \subseteq V$ that maximizes the density $f_w(S)$ (or equivalently $w(S)$) subject to the size constraint $|S| = k$. It is known that such a size restriction makes the problem much harder; in fact, the densest k-subgraph problem is NP-hard and the best known approximation ratio is $\Omega(1/n^{1/4+\varepsilon})$ for any $\varepsilon > 0$ (Bhaskara et al., 2010). The densest subgraph problem has also been extended to more general graph structures such as hypergraphs (Hu et al., 2017; Miyauchi et al., 2015) and multilayer networks (Galimberti et al., 2017). Moreover, to cope with the dynamics of real-world graphs and to model the limited computation resources in reality, some literature has considered dynamic settings (Epasto et al., 2015; Hu et al., 2017; Nasir et al., 2017) and streaming settings (Angel et al., 2012; Bahmani et al., 2012; Bhattacharya et al., 2015; McGregor et al., 2015), respectively. The average-degree density itself has also been generalized by modifying the numerator (Mitzenmacher et al., 2015; Miyauchi & Kakimura, 2018; Tsourakakis, 2015) or the denominator (Kawase & Miyauchi, 2018) of $d(S) = \frac{w(S)}{|S|}$, for some specific purposes.

B. Notation

We give the summary of notation in Table 4.

Table 4. Notation.

Notation	Description
$G = (V, E, w)$	Undirected graph
$E(S) = \{\{u,v\} \in E : u, v \in S\}$	Subset of edges induced by $S \subseteq V$
$G[S] = (S, E(S))$	Subgraph induced by S
$w : E \to \mathbb{R}_{>0}$	Expected edge weights
$w(S) = \sum_{e \in E(S)} w_e$	Sum of edge weights in $E(S)$
$f_w(S) = w(S)/\lvert S\rvert$	Degree density for weight w and $S \subseteq V$
$\mathbf{1}_{E(S)} \in \{0,1\}^E$	Indicator vector of $E(S)$
$\boldsymbol{x}_t = (\mathbf{1}_{E(S_1)}, \ldots, \mathbf{1}_{E(S_t)}) \in \{0,1\}^{E \times t}$	Sequence of indicator vectors
$(r_1(S_1), \ldots, r_t(S_t)) \in \mathbb{R}^t$	Sequence of observed rewards
$A_{\boldsymbol{x}_t} = \sum_{i=1}^{t} \mathbf{1}_{E(S_i)} \mathbf{1}_{E(S_i)}^\top + I$	Design matrix
$\boldsymbol{b}_{\boldsymbol{x}_t}$	$\boldsymbol{b}_{\boldsymbol{x}_t} = \sum_{i=1}^{t} \mathbf{1}_{E(S_i)} r_i(S_i) \in \mathbb{R}^E$
$\hat{\boldsymbol{w}}_t = A_{\boldsymbol{x}_t}^{-1} \boldsymbol{b}_{\boldsymbol{x}_t} \in \mathbb{R}^E$	Regularized least-square estimator
$\lVert x \rVert_B = \sqrt{x^\top B x}$	Quadratic (ellipsoidal) norm
$N(v) = \{u \in V : \{u,v\} \in E\}$	Set of neighbors of $v \in V$
$\deg_{\max} = \max_{v \in V} \lvert N(v)\rvert$	Maximum degree of vertices
$p = (p(S))_{S \subseteq S} \in \mathcal{P}$	Predetermined proportions of queries
$\Sigma_p = \sum_{S \subseteq S} p(S) \mathbf{1}_{E(S)} \mathbf{1}_{E(S)}^\top$	Design matrix for $p \in \mathcal{P}$
$\rho_p = \max_{x \in [-1,1]^m} \lVert x \rVert^2_{\Sigma_p^{-1}}$	Upper bound of maximal con dence width
$N_S(v) = \{u \in S : \{u,v\} \in E\}$	Set of neighboring vertices of v in $G[S]$
$E_S(v) = \{\{u,v\} \in E : u \in N_S(v)\}$	Set of incident edges to v in $G[S]$
$\hat{X}_F(k) = \frac{1}{k} \sum_{s=1}^{k} X_F(s)$	Empirical mean of weights for k samples
$\widehat{\deg}_{S,v}(t) = \hat{X}_{E_S(v)}(t)$	Empirical degree in $S \subseteq V$ for $v \in S$
$\hat{\rho}(S_{n\,t+1}) = \frac{\frac{1}{2} \sum_{v \in S_{n\,t+1}} \widehat{\deg}_{S_{n\,t+1}}(v;t)}{\lvert S_{n\,t+1}\rvert}$	Empirical quality function at round t

C. LP-based exact algorithm for the densest subgraph problem

We describe an exact algorithm for the densest subgraph problem based on the following LP and simple rounding procedure proposed by Charikar (2000) which we use in our proposed algorithm.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{e \in E} w_e x_e \\
\text{subject to} \quad & x_e \le y_u,\ x_e \le y_v \quad \forall e = \{u,v\} \in E, \\
& \sum_{v \in V} y_v = 1, \\
& x_e, y_v \ge 0 \qquad \forall e \in E, \forall v \in V.
\end{aligned} \tag{8}
$$

Let (x^*, y^*) be an optimal solution to the above LP. For a real number $r \ge 0$, the algorithm considers a sequence of subsets vertices $S(r) = \{v \in V : y^*_v \ge r\}$ and finds $r^* \in \operatorname{argmax}_{r \in [0,1]} f_w(S(r))$. Such an r^* can be obtained by simply examining $r = y^*_v$ for each $v \in V$. Finally, the algorithm outputs $S(r^*)$. Charikar (2000) proved that the output $S(r^*)$ is an optimal solution to the densest subgraph problem.

D. Arm allocation strategy

In this section, we introduce a possible allocation strategy that can be used in the DSG-LUCB algorithm. To reduce the number of samples, good arm allocation strategy makes con dence bound shrinking fast. We de ne the G-allocation for a set family $\mathcal{S} \subseteq 2^V$ as

$$
p^* = \operatorname*{argmin}_{p \in \mathcal{P}} \max_{S \in \mathcal{S}} \lVert \mathbf{1}_{E(S)} \rVert^2_{\Sigma_p^{-1}}.
$$

There are existing studies that proposed approximate solutions to solve it in the experimental design literature (Bouhtou et al., 2010; Sagnol, 2013); we can solve a continuous relaxation of the problem by a projected gradient algorithm when the support size $\lvert\mathcal{S}\rvert$ is not so large. For details on G-allocation or standard G-optimal design, see Soare et al. (2014) or see Pukelsheim (2006).

In general, an algorithm that adaptively changes an arm selection strategy based on the past observations at each round, which is called an *adaptive algorithm*, is desired because samples should be allocated for comparison of near-optimal arms

in order to reduce the number of samples. On the other hand, the algorithm that fixes all arm selections before observing any reward is called the *static algorithm*. Although the static algorithm is not able to focus on estimating near-optimal arms, it can be used to analyze the worst case optimality. In our text, each arm corresponds to an edge set; it is rare that any set is able to query since the possible choices are exponential. Therefore, we design a static algorithm DS-Uniform for solving Problem 1. On the other hand, if we are allowed to query any action as in Problem 2, we can design an adaptive algorithm DS-SR.

E. Technical lemmas for Theorem 1

We introduce random event E_t which characterizes the event that the confidence bounds of any feasible solution $S \subseteq V$ are valid at round t. We define a random event E_t as follows:

$$E_t = \{\forall S \subseteq V \text{ and } v \in S,\ |w(S) - w_t(S)| \leq C_t \|\mathbf{1}_{E(S)}\|_{A_{x_t}^{-1}}\}. \tag{9}$$

The following lemma states that event $E = \bigcap_{t=1}^{T} E_t$ occurs with high probability.

Lemma 1. The event E occurs with probability at least $1-\delta$.

The proof is omitted since it is straightforward from Proposition 1 and union bounds.

Lemma 2. For a fixed round $t > m$, assume that E_t occurs. Then, if the algorithm stops at round t, the output of the algorithm S_{OUT} satisfies $f_w(S^*) - f_w(S_{OUT}) \leq \varepsilon$.

Proof. If $S_{OUT} = S^*$, we obviously have the desired result. Then, we shall assume $S_{OUT} \neq S^*$.

$$\begin{aligned}
f_w(S_{OUT}) &\geq f_{w_t}(S_{OUT}) - \frac{C_t \|\mathbf{1}_{E(S_{OUT})}\|_{A_{x_t}^{-1}}}{|S_{OUT}|} \\
&\geq \max_{S \neq S_{OUT},\, S \subseteq V} f_{w_t}(S) + \frac{C_t Z_t}{2} \\
&\geq f_{w_t}(S^*) + \frac{\max_{x \in [-1,1]^m} \|x\|_{A_{x_t}^{-1}}}{2} \\
&\geq f_{w_t}(S^*) + \frac{\max_{S \subseteq V} \|\mathbf{1}_{E(S)}\|_{A_{x_t}^{-1}}}{2} \\
&\geq f_{w_t}(S^*) + \frac{C_t \|\mathbf{1}_{E(S^*)}\|_{A_{x_t}^{-1}}}{|S^*|} - \varepsilon \\
&\geq f_w(S^*) - \varepsilon,
\end{aligned}$$

where the first and last inequalities follow from the event E_t, and the second inequality follows from the stopping condition, and the third inequality follows from the definition of Z_t and approximation ratio. □

In Miyauchi & Takeda (2018), they provided the following lemma, which we also use to prove Theorem 1.

Lemma 3 (Miyauchi & Takeda (2018), Lemma 2). Let $G = (V, E)$ be an undirected graph. Let w_1 and w_2 be edge-weight vectors such that $\|w_1 - w_2\|_1 \leq \varepsilon$ holds for $\varepsilon > 0$. Then, for any $S \subseteq V$, it holds that

$$|f_{w_1}(S) - f_{w_2}(S)| \leq \frac{\sqrt{m}}{2}\varepsilon.$$

F. Proof of Theorem 1

Proof. We know that the event E holds with probability at least $1-\delta$ from Lemma 1. Therefore, we only need to prove that, under event E, the algorithm returns a set whose density is at least $f_w(S^*) - \varepsilon$ and provide an upper bound of number of queries. From Lemma 2, on the event E, the algorithm outputs $S_{OUT} \subseteq V$ that satisfies $f_w(S^*) - f_w(S_{OUT}) \leq \varepsilon$.

Next, we focus on bounding the number of queries. Recall that Algorithm 2 employs a stopping condition:

$$f_{w_t}(S_t) \quad \frac{C_t \| E(S_t) \|}{|S_t|} \quad \max_{S \in \mathcal{S}_t : S \subseteq V} f_{w_t}(S) + \frac{C_t Z_t}{2} \quad ; \tag{10}$$

where Z_t denotes the objective of the approximate solution to P1. A sufficient condition of the stopping condition is that for S and for $t > m$,

$$f_{w_t}(S) \quad \frac{C_t \| E(S) \|}{|S|} \quad \max_{S \in \mathcal{S} : S \subseteq V} f_{w_t}(S) + \frac{C_t Z_t}{2} \quad ; \tag{11}$$

Since $Z_t \quad \max_{x \in [-1,1]^m} \|x\|_{A_{x_t}^{-1}}$ and $\| E(S) \| \quad \max_{x \in [-1,1]^m} \|x\|_{A_{x_t}^{-1}}$, the following inequality is also a sufficient condition to stop:

$$f_{w_t}(S) \quad \frac{C_t \max_{x \in [-1,1]^m} \|x\|_{A_{x_t}^{-1}}}{|S|} \quad \max_{S \in \mathcal{S} : S \subseteq V} f_{w_t}(S) + \frac{C_t \max_{x \in [-1,1]^m} \|x\|_{A_{x_t}^{-1}}}{2} \quad : \tag{12}$$

Recall that $T_t(S)$ be the number of times that $S \subseteq S$ is queried before t-th round in the algorithm. We denote p_{x_t} by $p_{x_t} = (T_t(S)/t)_{S \subseteq S}$. From the above definitions, the design matrix is $A_{x_t} = t \quad p_{x_t}$. Recall that $_p = \max_{x \in [-1,1]^m} \|x\|_{p^{-1}}^2$, and let $_{p_{x_t}} = \max_{x \in [-1,1]^m} \|x\|^2_{p_{x_t}^{-1}}$. From the fact that $\lim_{t \to \infty} \quad p_{x_t} = \quad p$, for sufficiently large t m we have that $| \quad p \quad p_{x_t} |$ with probability at least $1 \quad \frac{}{2}$ where $2 (0,1)$ and > 0. Let $S = \operatorname{argmax}_{S \in \mathcal{S} : S \subseteq V} f_{w_t}(S)$. Then, (12) is rewritten as follows:

$$f_{w_t}(S) \quad f_{w_t}(S) \quad \frac{1}{|S|} + \frac{1}{2} \quad C_t \sqrt{\frac{ \quad p}{t}} \quad : \tag{13}$$

Next, we show the following inequality.

$$f_{w_t}(S) \quad f_{w_t}(S) \quad \min \quad \sqrt{2m} C_t \sqrt{\frac{ \quad p}{t}} : \tag{14}$$

From Proposition 1, with probability $1 \quad \frac{}{2}$, we have $\|w \quad w_t\|_1 \quad C_t \max_S \| \quad E(S) \|_{A_{x_t}^{-1}}$. From Lemma 3, we see that $f_{w_t}(S) \quad f_w(S) \quad \sqrt{\frac{m}{2}} C_t \max_{S \subseteq V} \| \quad E(S) \|_{A_{x_t}^{-1}}$ and $f_w(S) \quad f_{w_t}(S) \quad \sqrt{\frac{m}{2}} C_t \max_{S \subseteq V} \| \quad E(S) \|_{A_{x_t}^{-1}}$. Therefore, for sufficiently large t m such that $| \quad p \quad p_{x_t} |$ with probability at least $1 \quad \frac{}{2}$, we have that

$$\begin{aligned}
f_{w_t}(S) \quad f_{w_t}(S) \quad f_w(S) \quad f_w(S) & \quad \sqrt{2m} C_t \max_{S \subseteq V} \| \quad E(S) \|_{A_{x_t}^{-1}} \\
f_w(S) \quad f_w(S) & \quad \sqrt{2m} C_t \max_{x \in \{-1,1\}^m} \|x\|_{A_{x_t}^{-1}} \\
= f_w(S) \quad f_w(S) & \quad \sqrt{2m} C_t \sqrt{\frac{p_{x_t}}{t}} \\
f_w(S) \quad f_w(S) & \quad \sqrt{2m} C_t \sqrt{\frac{ \quad p}{t}} \\
\min & \quad \sqrt{2m} C_t \sqrt{\frac{ \quad p}{t}} :
\end{aligned}$$

Then, we obtain (14). Combining (13) and (14), we see that the following inequality is a sufficient condition to stop:

$$\min \quad \sqrt{2m} C_t \sqrt{\frac{ \quad p}{t}} \quad \frac{1}{|S|} + \frac{1}{2} \quad C_t \sqrt{\frac{ \quad p}{t}} \quad : \tag{15}$$

Solving (15) with respect to t, we obtain

$$t \quad (\sqrt{2m} + |S| \quad ^{-1} + 2 \quad ^{-1})^2 C_t^2 H \; ; \tag{16}$$

where recall that H is defined as

$$H = \frac{}{(\ _{\min} + \ )^2}:$$

Therefore, from the above, we obtain in $\sqrt{2m} + |S| \ ^1 + 2 \ ^1 \ ^2 C_t^2 H$ as a sufficient condition to stop. Let $> m$ be the stopping time of the algorithm. From the above discussion, since $|S| \geq 2$, we see that

$$\sqrt{2m} + \frac{+1}{2} \ ^2 C^2 H : \tag{17}$$

Now we bound C in (17). We have $\det(A_x) \quad (+)^m$, which is obtained by Lemma 10 in Abbasi-Yadkori et al. (2011). Then, we obtain the following inequality in the similar manner in Theorem 2 in Xu et al. (2018):

$$\begin{aligned} C \quad & R^0 \ 2\log\frac{\det(A_x)^{\frac{1}{2}}\det(\ I\ )^{\frac{1}{2}}}{} + \ ^{\frac{1}{2}}L \\ & R^0 \ 2\log\frac{1}{} + m\log\ 1 + \ - \ + \ ^{\frac{1}{2}}L: \end{aligned} \tag{18}$$

Using (18), we give an upper bound of . We also use a similar proof strategy as in Xu et al. (2018). First, let us consider the case $> 4m(\sqrt{m} + \sqrt{2})^2 R^{\alpha} H$, where recall that $R^0 = \sqrt{\deg_{\max}} R$. Using the facts that $(a + b)^2 \quad 2(a^2 + b^2)$ for $a; b > 0$ and $\log(1 + x) \quad x$, we have

$$\sqrt{2m} + \frac{+1}{2} \ ^2 C^2 H \quad 2 \sqrt{2m} + \frac{+1}{2} \ ^2 H \quad 4R^{\alpha}\log\frac{1}{} + \frac{R^{\alpha} m}{} \ + L \ ^2 :$$

Thus, we obtain

$$\begin{aligned} & 1 \quad 2 \sqrt{2m} + \frac{+1}{2} \ ^2 \frac{R^{\alpha} m H}{} \ ^{!} \ ^1 (8H \ R^{\alpha}\log 1= \ + 2H \ L \ ^2) \\ & 2(8H \ R^{\alpha}\log 1= \ + 2H \ L \ ^2); \end{aligned}$$

where the last inequality holds from $> 4m(\sqrt{m} + \sqrt{2})^2 R^{\alpha} H$ and $= \sqrt{\frac{4}{7}}$. Therefore, in this case, we obtain

$$= O \quad \deg_{\max}^2 R^2 \log\frac{1}{} + L \ ^2 \quad H \quad :$$

Second, we consider $\frac{R^{\alpha}}{L^2}\log\ ^{\frac{1}{}}$. From this bound and (18), we have

$$C \quad 2R^0 \ 2\log\frac{1}{} + m\log\ 1 + \ - \ :$$

Let $V(m; \ ) = \sqrt{2m} + \frac{+1}{2} \ ^2$. Therefore, we obtain

$$V(m; \ )C^2 H \quad 4V(m; \ )R^{\alpha} \ 2\log\frac{1}{} + m\log\ 1 + \ - \quad H :$$

Let $N = 8V(m; \ )R^{\alpha}\log\frac{1}{}H$ and let ^{0} be a parameter satisfying:

$$= N + 4V(m; \ )R^{\alpha} m\log\ 1 + \frac{}{^0} \ H : \tag{19}$$

It is easy to see $\varepsilon' \le \varepsilon$. Then, we have

$$\varepsilon' = N + 4V(m,\delta)R^2 m \log\left(1 + \frac{\varepsilon'}{H}\right) \le N + 4V(m,\delta)R^2 m \frac{\varepsilon'}{H},$$

where the second inequality follows from the fact $\log(1+x) \le \sqrt{x}$. Solving this quadratic inequality for $\sqrt{\varepsilon'}$, we have

$$\sqrt{\varepsilon'} \le \frac{2V(m,\delta)R^2 mH}{\sqrt{\varepsilon'}} + \sqrt{\frac{4V(m,\delta)^2 R^4 m^2 H^2}{\varepsilon'} + N} \le 2\sqrt{\frac{4V(m,\delta)^2 R^4 m^2 H^2}{\varepsilon'} + N}. \tag{20}$$

Let $M = 2\sqrt{\frac{4V(m,\delta)^2 R^4 m^2 H^2}{\varepsilon'} + N}$. We can give an upper bound by (19) and (20) as follows:

$$\varepsilon' \le 8V(m,\delta)R^2 H \log\frac{1}{\delta} + 4H\, V(m,\delta)R^2 m \log\left(1 + \frac{M}{H}\right).$$

Note that $\log M = O\left(\log\left(R^4 m^4 H^2 + mR^2 \log\frac{1}{\delta} H\right)\right)$ since $V(m,\delta) = O(m)$. From the above and $R' = \sqrt{\deg_{\max}} R$, we obtain

$$\begin{aligned}
&= O\left(m\deg_{\max} R^2 H \log\frac{1}{\delta} + m\deg_{\max} R^2 H \log\left(\deg^2_{\max} R^4 m^4 H^2 + m\deg_{\max} R^2 H \log\frac{1}{\delta}\right)\right) \\
&= O\left(m\deg_{\max} R^2 H \log\frac{1}{\delta} + C(H,\delta)\right)
\end{aligned}$$

where

$$\begin{aligned}
C(H,\delta) &= m\deg_{\max} R^2 H \log\left(\deg^2_{\max} R^4 m^4 H^2 + m\deg_{\max} R^2 H \log\frac{1}{\delta}\right) \\
&= O\left(m\deg_{\max} R^2 H \log\left(\deg_{\max} RmH \log\frac{1}{\delta}\right)\right).
\end{aligned}$$

□

G. Technical lemmas for Theorem 2

First we introduce a standard concentration inequality of sub-Gaussian random variables (Chen et al., 2014).

Lemma 4 (Chen et al. (2014), Lemma 6). Let $X_1, \dots, X_k$ be k independent random variables such that, for each $i \in [k]$, random variable $X_i - E[X_i]$ is R-sub-Gaussian distributed, i.e., $\forall a \in \mathbb{R}$, $E[\exp(a(X_i - E[X_i]))] \le \exp(R^2 a^2/2)$. Let $\bar{X} = \frac{1}{k}\sum_{i=1}^{k} X_i$ denote the average of these random variables. Then, for any $\varepsilon > 0$, we have

$$\Pr\left[\left|\bar{X} - E[\bar{X}]\right| \ge \varepsilon\right] \le 2\exp\left(-\frac{k\varepsilon^2}{2R^2}\right).$$

For $S \subseteq V$, $v \in S$, and expected weight w, we denote by $\deg_S(v)$ the weighted degree of $v \in S$ on $G[S]$ in terms of the true edge weight w. We show the following lemma used for analysis of Algorithm 3.

Lemma 5. Given an phase $t \in \{1, \dots, n-1\}$, we define random event

$$E_t = \left\{\forall v \in S_{n-t+1},\ \deg_{S_{n-t+1}}(v) \ge \frac{1}{2}\deg_{S_{n-t+1}}(v^*_{n-t+1}; t)\right\}. \tag{21}$$

Then, we have

$$\Pr\left[\bigcap_{t=1}^{n-1} E_t\right] \ge 1 - \frac{2\deg_{\max} 2^n R^2}{\delta^2}(n+1)^3 \exp\left(-\frac{\left(\sum_{i=1}^{n+1} T_i\right)^2}{4n^2\deg_{\max}\log\frac{n-1}{\delta}}\right). \tag{22}$$

Proof. For any $S \subseteq V$, and $v \in S$, recall that $X_{E_S(v)}(i)$ is i-th observation of edge-weight $E_S(v)$ for $i \in [k]$. Then, $X_{E_S(v)}(i) - E[X_{E_S(v)}(i)]$ follows a $(\sqrt{|E_S(v)|}R)$-sub-Gaussian distribution. We can assume that the sequence of weights for each subset of edges is drawn before the beginning of the game. Thus, $\hat{X}_{E_S(v)}(k)$ is well defined even if $E_S(v)$ has not been actually sampled k times. Therefore, from Lemma 4, for any $\gamma > 0$ we have that

$$\Pr\left[E[\hat{X}_{E_S(v)}(k)] - \hat{X}_{E_S(v)}(k) \geq \gamma\right] \leq 2\exp\left(-\frac{k\gamma^2}{2|E_S(v)|R^2}\right) \leq 2\exp\left(-\frac{k\gamma^2}{2\deg_{\max}R^2}\right). \tag{23}$$

Fix $t \in \{1, \ldots, n-1\}$ and x a vertex $v \in S_{n-t+1}$ in a phase t. If $|N_{S_{n-t+1}}(v)| = 0$, it is obvious that $\deg_{S_{n-t+1}}(v) = \widetilde{\deg}_{S_{n-t+1}}(v) = 0$. Therefore, we will consider a vertex $v \in S_{n-t+1}$ such that $|N_{S_{n-t+1}}(v)| \geq 1$ in the rest of the proof. By the definition of T_t for $t \in [1, 2, \ldots, n-1]$, we have

$$T_t = \frac{T\sum_{i=1}^{n+1} i}{(n-t)\log(n-1)} \geq \frac{T\sum_{i=1}^{n+1} i}{n\log(n-1)}.$$

Then for $T_{E_{S_{n-t+1}}(v)}(t)$, we have:

$$T_{E_{S_{n-t+1}}(v)}(t) = \sum_{i=1}^{t} \tau_i = \sum_{i=1}^{t}(T_i^0 - T_{i-1}^0) = T_t^0 \geq \frac{T_t}{2|S_{n-t+1}|} \geq \frac{T\sum_{i=1}^{n+1} i}{2n^2\log(n-1)}. \tag{24}$$

Let k^0 be the RHS of (24), i.e. $k^0 = \frac{T\sum_{i=1}^{n+1} i}{2n^2\log(n-1)}$. For $v \in S_{n-t+1}$, we have

$$\begin{aligned}
&\Pr\left[\deg_{S_{n-t+1}}(v) \neq \widetilde{\deg}_{S_{n-t+1}}(v)\right] \\
&= \Pr\left[E[\hat{X}_{E_{S_{n-t+1}}(v)}(T_{E_{S_{n-t+1}}(v)}(t))] \neq \hat{X}_{E_{S_{n-t+1}}(v)}(T_{E_{S_{n-t+1}}(v)}(t))\right] \\
&\leq \Pr\left[\exists S \subseteq V, u \in S : E[\hat{X}_{E_S(u)}(T_{E_{S_{n-t+1}}(v)}(t))] \neq \hat{X}_{E_S(u)}(T_{E_{S_{n-t+1}}(v)}(t))\right] \\
&\leq \sum_{S \supseteq v}\sum_{u \in S}\sum_{k=k^0}^{t} \Pr\left[E[\hat{X}_{E_S(u)}(k)] \neq \hat{X}_{E_S(u)}(k)\right] \\
&\leq \sum_{S \supseteq v}\sum_{u \in S}\sum_{k=k^0}^{t} 2\exp\left(-\frac{k\gamma^2}{2\deg_{\max}R^2}\right) \\
&= \sum_{S \supseteq v}\sum_{u \in S} \frac{2\exp\left(-\frac{k^0\gamma^2}{2\deg_{\max}R^2}\right)}{\exp\left(\frac{\gamma^2}{2\deg_{\max}R^2}\right) - 1} \\
&\leq \sum_{S \supseteq v}\sum_{u \in S} \frac{2\exp\left(-\frac{k^0\gamma^2}{2\deg_{\max}R^2}\right)}{\frac{\gamma^2}{2\deg_{\max}R^2}} \\
&= \sum_{S \supseteq v}\sum_{u \in S} \frac{4\deg_{\max}R^2}{\gamma^2}\exp\left(-\frac{(T\sum_{i=1}^{n+1} i)\gamma^2}{4n^2\deg_{\max}R^2\log(n-1)}\right) \\
&\leq \frac{4\deg_{\max}2^n nR^2}{\gamma^2}\exp\left(-\frac{(T\sum_{i=1}^{n+1} i)\gamma^2}{4n^2\deg_{\max}R^2\log(n-1)}\right);
\end{aligned} \tag{25}$$

where the third inequality follows by (23) and the fourth inequality follows by $e^x - 1 \geq x$.

Now using (25) and taking a union bound for all $t \in \{1, \dots, n-1\}$ and all $v \in S_{n-t+1}$, we obtain

$$\begin{aligned}
\Pr\left[\bigcap_{t=1}^{n-1} E_t^0\right] &= 1 - \Pr\left[\exists t \in \{1,\dots,n-1\}, v \in S_{n-t+1} : \deg_{S_{n-t+1}}(v) \le \tfrac{1}{2}\deg_{S_{n-t+1}}(v,t)\right] \\
&\ge 1 - \sum_{t=1}^{n-1} \sum_{v \in S_{n-t+1}} \Pr\left[\deg_{S_{n-t+1}}(v) \le \tfrac{1}{2}\deg_{S_{n-t+1}}(v,t)\right] \\
&\ge 1 - \sum_{t=1}^{n-1} \sum_{v \in S_{n-t+1}} \frac{4\deg_{\max} 2^n nR^2}{2} \exp\left(-\frac{(T - \sum_{i=1}^{n+1} i)^2}{4n^2 \deg_{\max} R^2 \log(n-1)}\right) \\
&= 1 - \frac{4\deg_{\max} 2^n nR^2}{2} \sum_{t=1}^{n-1} |S_{n-t+1}| \exp\left(-\frac{(T - \sum_{i=1}^{n+1} i)^2}{4n^2 \deg_{\max} R^2 \log(n-1)}\right) \\
&= 1 - \frac{4\deg_{\max} 2^n nR^2}{2} \sum_{t=1}^{n-1} (n-t+1) \exp\left(-\frac{(T - \sum_{i=1}^{n+1} i)^2}{4n^2 \deg_{\max} R^2 \log(n-1)}\right) \\
&= 1 - \frac{2\deg_{\max} 2^n R^2}{2} n^2(n+1) \exp\left(-\frac{(T - \sum_{i=1}^{n+1} i)^2}{4n^2 \deg_{\max} R^2 \log(n-1)}\right) \\
&\ge 1 - \frac{2\deg_{\max} 2^n R^2}{2} (n+1)^3 \exp\left(-\frac{(T - \sum_{i=1}^{n+1} i)^2}{4n^2 \deg_{\max} R^2 \log(n-1)}\right).
\end{aligned}$$

□

H. Proof of Theorem 2

Proof. First, we verify that the algorithms requires at most T queries. In each phase t, the number of samples Algorithm 4 requires is at most $T_t + |S_{n-t+1}|$, since we have that

$$\sum_{v \in S_{n-t+1}} \sum_{i=1}^{t} \left(T_i^0 - T_{i-1}^0\right) = \sum_{v \in S_{n-t+1}} T_t^0 \le \sum_{v \in S_{n-t+1}} \left(\frac{T_t}{|S_{n-t+1}|} + 1\right) \le T_t + |S_{n-t+1}|.$$

Therefore, the total number of queries used by the algorithm is bounded by

$$\begin{aligned}
\sum_{t=1}^{n-1} \left(T_t + |S_{n-t+1}|\right) &\le \sum_{t=1}^{n-1} T_t + \sum_{t=1}^{n-1} (n-t+1) \\
&\le \sum_{t=1}^{n-1} \left(\frac{T - \sum_{i=1}^{n+1} i}{(n-t)\log(n-1)} + 1\right) + \sum_{t=1}^{n-1} (n-t+1) \\
&\le \sum_{t=1}^{n-1} \frac{T - \sum_{i=1}^{n+1} i}{(n-t)\log(n-1)} + \sum_{i=1}^{n+1} i \\
&= \frac{(T - \sum_{i=1}^{n+1} i)}{\log(n-1)} \log(n-1) + \sum_{i=1}^{n+1} i \\
&= T - \sum_{i=1}^{n+1} i + \sum_{i=1}^{n+1} i = T.
\end{aligned}$$

Lemma 5 implies that the random event $E^0 := \bigcap_{t=1}^{n-1} E_t^0$ occurs with probability at least $1 - \frac{2\deg_{\max} 2^n R^2}{2}(n+1)^3 \exp\left(-\frac{(T - \sum_{i=1}^{n+1} i)^2}{4n^2 \deg_{\max} R^2 \log(n-1)}\right)$. We shall assume the event E^0 occurs in the rest of the proof, because we only need to show that the algorithm outputs a solution S_{OUT} that guarantees $f_w(S_{\text{OUT}}) \ge \frac{f_w(S^*)}{2}$ under E^0.

Let $S \subseteq V$ be an optimal solution in terms of the expected weight. Choose an arbitrary vertex $v \in S$. From the optimality of $S \subseteq V$, it holds that

$$f_w(S) = \frac{w(S)}{|S|} \geq \frac{w(S \setminus \{v\})}{|S| - 1} = f_w(S \setminus \{v\}).$$

By using the fact that $w(S \setminus \{v\}) = w(S) - \deg_S(v)$, the above inequality can be transformed into

$$\deg_S(v) \geq f_w(S). \tag{26}$$

Let $S \subseteq V$ be the last subset over the phases that satisfies $S \neq \emptyset$ and let $t \in [1, \ldots, n]$ be its phase. Let S_{OUT} be the phase t such that $S_{n-t+1} = S_{\mathrm{OUT}}$. Then we have

$$f_w(S_{\mathrm{OUT}}) = \frac{\frac{1}{2}\sum_{v \in S_{\mathrm{OUT}}} \deg_{S_{\mathrm{OUT}}}(v)}{|S_{\mathrm{OUT}}|} \geq \frac{\frac{1}{2}\sum_{v \in S_{\mathrm{OUT}}} \deg_{S_{\mathrm{OUT}}}(v, \mathrm{OUT})}{|S_{\mathrm{OUT}}|} \geq \frac{\frac{1}{2}\sum_{v \in S} \deg_S(v, \mathrm{OUT})}{|S|} \geq \frac{1}{2},$$

where the first inequality follows from event E^0, the second inequality follows from the greedy choice of S_{OUT}. Recall that the algorithm removes the vertex that satisfies $v \in \operatorname{argmin}_{v \in S} \deg_S(v, \mathrm{OUT})$ in the phase. Therefore, from the definition of S, it is clear that $v \in S$. Using this property, we further have that

$$\begin{aligned}
\frac{\frac{1}{2}\sum_{v \in S} \deg_S(v, \mathrm{OUT})}{|S|} - \frac{1}{2} &\geq \frac{\frac{1}{2}\sum_{v \in S} \deg_S(v, \mathrm{OUT})}{|S|} - \frac{1}{2} \\
&= \frac{1}{2}\deg_S(v, \mathrm{OUT}) - \frac{1}{2} \\
&\geq \frac{1}{2}\deg_S(v) - \frac{1}{2} \\
&\geq \frac{1}{2} f_w(S) - \frac{1}{2},
\end{aligned}$$

where the second inequality follows from event E^0, and third inequality follows from the fact $S \subseteq S$, and the last inequality follows from the fact $v \in S$ and inequality (26). Therefore, we obtain $f_w(S_{\mathrm{OUT}}) \geq \frac{1}{2} f_w(S) - \frac{1}{2}$. That concludes the proof. □

I. Details of experiments for DS-Lin

I.1. Behavior of DS-Lin

We first analyze the behavior of our proposed algorithm with respect to the number of iterations. In the previous section, we confirmed that the solution obtained after 10,000 iterations is almost densest in terms of unknown k. A natural question here is how the density of solutions approaches to such a sufficiently large value. In other words, does our algorithm is sensitive to the choice of the number of iterations? In this section, we answer these questions by conducting the following experiments. We terminate the while-loop of our algorithm once the number of iterations exceeds $\{100, 200, \ldots, 10{,}000\}$, and follow the density values of solutions in terms of w. For each instance, we again run our algorithm for ten times, and report the average value.

The results are shown in Figure 2. As can be seen, as the number of iterations increases, the density value converges to the sufficiently large value (close to the optimum). Although the density value sometimes drops down, the decrease is quite small.

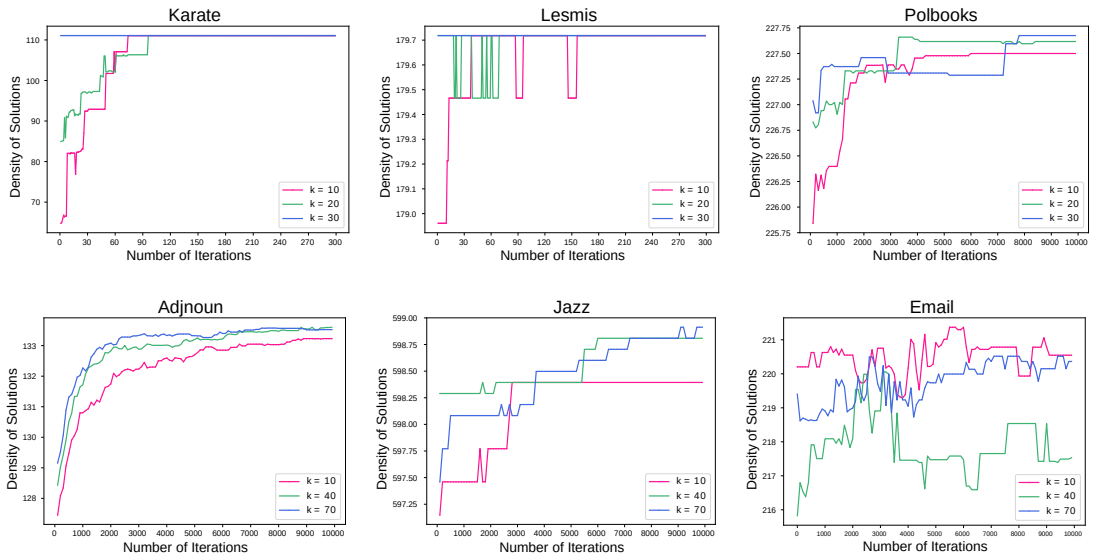

Figure 2. Results for the behavior of our proposed algorithm with respect to the number of iterations. Each point is an average over 10 runs of the algorithm.

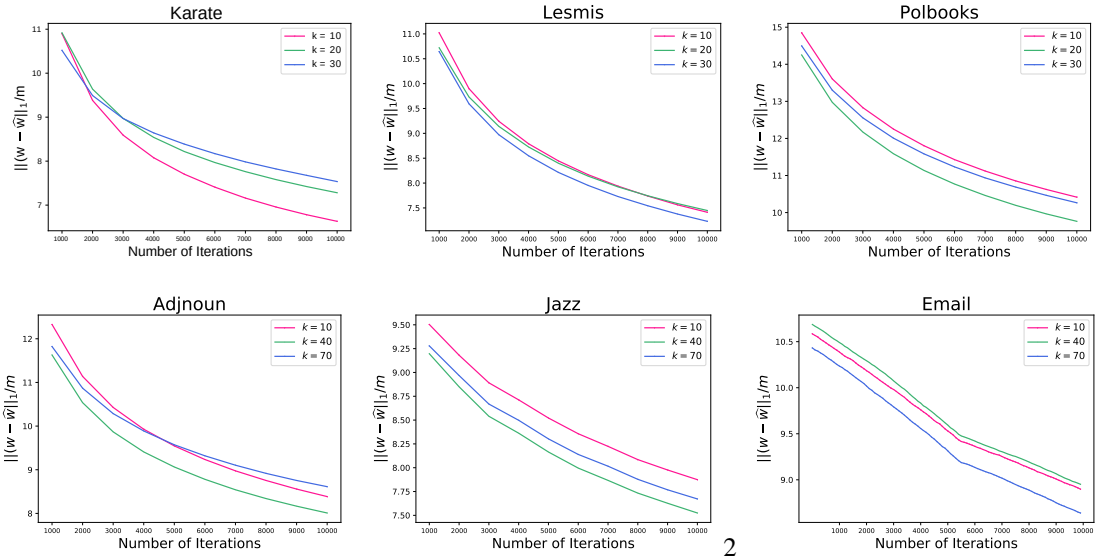

Figure 3. Results for the estimation of the expected weight w. Each point is an average over 10 runs of the algorithm.

I.2. Estimation of the expected weight

We next explain the reason why our proposed algorithm DS-Lin performs fairly well. To this end, we focus on the quality of the estimated edge weight obtained by the algorithm. We measure the quality of the estimated edge weight $\hat{w}_t$ by comparing with the expected weight w; specifically, we compute $\|w - \hat{w}_t\|_1/m$. The experimental setup is exactly the same as that in the previous section.

The results are depicted in Figure 3. As can be seen, as the number of iterations increases, $\hat{w}_t$ converges to the true edge weight w. It is very likely that the high performance of our algorithm is derived from the high-quality estimation of the expected edge weight w.

Algorithm 7 Robust optimization with oracle intervals (R-Oracle)

Input : Graph $G = (V; E)$, oracle intervals $W = \ _{e2E}[l_e; r_e]$ where $l_e = \min\{w_e \ \ 1; 0\}$ and $r_e = w_e + 1$, sampling oracle, $\ \ 2 \ (0; 1)$, and $" > 0$
Output : (S_{out})
for *each* $e \ 2 \ E$ **do**
  if $l_e = r_e$ **then**
    $l_e^{\text{out}} \quad l_e; r_e^{\text{out}} \quad r_e;$
  end
  else
    $S_w \quad$ Output of Charikar's LP-based exact algorithm for $G(V; E; w \ );$
    $t_e \quad \frac{m(r_e \ \ l_e)^2 \ln \frac{2m}{}}{"^2 f_w \ (S_w \ )^2} \ ;$
    Sample e for t_e times;
    $\hat{p}_e \quad \hat{X}_e(t_e);$
    $\frac{"f_w p \frac{(S_w \ )}{2m}}{};$
    $l_e^{\text{out}} \quad \max\{l_e; \hat{p}_e \quad g$ and $r_e^{\text{out}} \quad \min\{r_e; \hat{p}_e + \ g;$
  end
end
$W_{\text{out}} \quad _{e2E}[l_e^{\text{out}}; r_e^{\text{out}}];$
$S_{\text{out}} \quad$ Output of Charikar's LP-based exact algorithm for $G(V; E; l_{\text{out}});$
return $S_{\text{out}};$

J. Details of experiments for DS-SR

J.1. Description of R-Oracle

We describe the entire procedure of R-Oracle in Algorithm 7. This algorithm employs the robust optimization model proposed by Miyauchi & Takeda (2018). Their robust optimization model takes intervals of edge weights as its input. We generate the intervals $W = \ _{e2E}[l_e; r_e]$ based on unknown edge weight w, i.e., $l_e = \min\{w_e \ \ 1; 0\}$ and $r_e = w_e + 1$. Algorithm 7 first obtains the optimal solution $S_w \ $ in terms of extreme edge weight $w \ = (l_e)_{e2E}$ and computes the value of $f_w \ (S_w \ )$. Then, for each single edge $e \ 2 \ E$, the algorithm calls the sampling oracle for an appropriate number of times and obtains the empirical mean. Using the empirical means, the algorithm constructs intervals $W_{\text{out}} \quad _{e2E}[l_e^{\text{out}}; r_e^{\text{out}}]$, and computes a densest subgraph S_{out} on G with $w_{\text{out}} = (l_e^{\text{out}})_{e2E}$.

J.2. The number of samples for single edges in DS-SR

We report experimental results on the size of queried edge subsets in DS-SR (cumulative) over 100 runs for all instances in Figure 4.

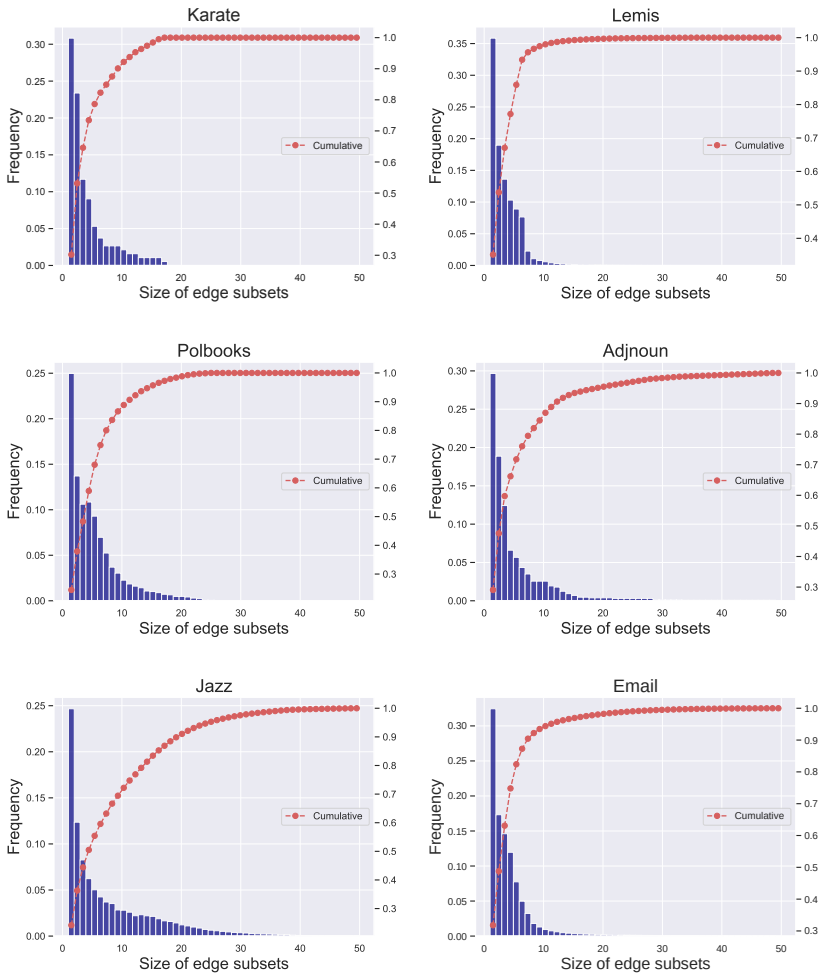

Figure 4. Fraction of the size of queried edge subsets in DS-SR (cumulative) over 100 runs.