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 $\Omega(1/n^{1/4+\epsilon})$ for any $\epsilon > 0$ (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 ϵ-optimal solution with probability at least $1-\delta$ for $\epsilon > 0$ and $\delta \in (0, 1)$ (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 ϵ-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 $\frac{1}{2}\mathrm{OPT} - \epsilon$, 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 $n = |V|$ vertices and $m = |E|$ 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 $f_w(S) = w(S)/|S|$, 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 ϕ_e over reals. $w : E \to \mathbb{R}_{>0}$ is the expected edge weights, where $w(e) = \mathbb{E}_{X\sim\phi_e}[X]$. 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 ϕ_e for $e \in E$, then for all $r \in \mathbb{R}$, X satisfies $\mathbb{E}[\exp(rX - r\mathbb{E}[X])] \leq \exp(R^2r^2/2)$. We define the optimal solution as $S^* = \mathrm{argmax}_{S\subseteq V} f_w(S)$.

We first address the setting in which the learner can stop the game at any round if she can return an ϵ-optimal solution for $\epsilon > 0$ 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 $\epsilon > 0$ be a required accuracy and $\delta \in (0, 1)$ be a confidence level. Then, the goal is to find $S_{\mathtt{OUT}} \subseteq V$ that satisfies $\Pr[f_w(S^*) - f_w(S_{\mathtt{OUT}}) \leq \epsilon] \geq 1 - \delta$, 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 $S_{\mathtt{OUT}} \subseteq V$ that maximizes $f_w(S_{\mathtt{OUT}})$ 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 $\Pr[f_w(S^*) - f_w(S_{\mathtt{OUT}}) \leq \epsilon] \geq 1 - \delta$. 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 $\boldsymbol{\chi}_{E(S)} \in \{0, 1\}^E$ be the indicator vector of $E(S) \subseteq E$, i.e., for each $e \in E$, $\boldsymbol{\chi}_{E(S)}(e) = 1$ if $e \in E(S)$ and $\boldsymbol{\chi}_{E(S)}(e) = 0$ 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 $\mathbf{x}_t = (\boldsymbol{\chi}_{E(S_1)}, \ldots, \boldsymbol{\chi}_{E(S_t)}) \in \{0, 1\}^{E\times t}$ 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 $A_{\mathbf{x}_t}$ as

$$A_{\mathbf{x}_t} = \sum_{i=1}^{t} \boldsymbol{\chi}_{E(S_i)}\boldsymbol{\chi}_{E(S_i)}^\top + \lambda I \in \mathbb{R}^{E\times E}$$

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

$$\widehat{w}_t = A_{\mathbf{x}_t}^{-1} b_{\mathbf{x}_t} \in \mathbb{R}^E. \tag{1}$$

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 $\|x\|_B = \sqrt{x^\top Bx}$. 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 $\widehat{w}_t = A_{\mathbf{x}_t}^{-1} b_{\mathbf{x}_t}$, where $\mathbf{x}_t$ 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 $\mathbf{x}_t$, with probability at least $1 - \delta$, the inequality*

$$|w(S) - \widehat{w}_t(S)| \leq C_t \|\boldsymbol{\chi}_{E(S)}\|_{A_{\mathbf{x}_t}^{-1}} \tag{2}$$

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

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

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

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

$$f_{\widehat{w}_t}(\widehat{S}_t) - \frac{C_t \|\boldsymbol{\chi}_{E(\widehat{S}_t)}\|_{A_{\mathbf{x}_t}^{-1}}}{|\widehat{S}_t|} \geq \max_{S \subseteq V \,:\, S \neq \widehat{S}_t} f_{\widehat{w}_t}(S) + \frac{C_t \max_{S \subseteq V} \|\boldsymbol{\chi}_{E(S)}\|_{A_{\mathbf{x}_t}^{-1}}}{|S|} - \epsilon.$$

The above stopping condition guarantees that the output satisfies $f_w(S^*) - f_w(S_{\mathtt{OUT}}) \leq \epsilon$ with probability at least $1-\delta$. However, computing $\max_{S \subseteq V} \|\boldsymbol{\chi}_{E(S)}\|_{A_{\mathbf{x}_t}^{-1}}$ 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:

$$\text{P1: max. } \|x\|_{A_{\mathbf{x}_t}^{-1}} \text{ s.t. } -e \leq x \leq e, \tag{4}$$

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

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

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

where $Q = (q_{ij}) \in \mathbb{R}^{m \times m}$ is a given symmetric matrix. Ye (1999) modified 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 confidence bound $\max_{S \subseteq V} \|\boldsymbol{\chi}_{E(S)}\|_{A_{\mathbf{x}_t}^{-1}}$ 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 $\max_{S \subseteq V} \|\boldsymbol{\chi}_{E(S)}\|^2_{A_{\mathbf{x}_t}^{-1}}$. Therefore, using Algorithm 1, we can ensure the following stopping condition in polynomial time:

$$f_{\widehat{w}_t}(\widehat{S}_t) - \frac{C_t \|\boldsymbol{\chi}_{E(\widehat{S}_t)}\|_{A_{\mathbf{x}_t}^{-1}}}{|\widehat{S}_t|} \geq \max_{S \subseteq V \,:\, S \neq \widehat{S}_t} f_{\widehat{w}_t}(S) + \frac{C_t Z_t}{2\alpha} - \epsilon, \tag{6}$$

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 semidefinite 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):

$$\text{QP: max. } \sum_{1 \leq i,j \leq m} q_{ij} x_i x_j \text{ s.t. } -e \leq x \leq e,$$

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

Proposed algorithm. Let $T_t(S)$ be the number of times that $S \subseteq \mathcal{S}$ 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 defined as follows. Let $\mathcal{P}$ be a $|\mathcal{S}|$-dimensional probability simplex. We define 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 p, 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 $A_{\mathbf{x}_t}$ and $b_{\mathbf{x}_t}$, and also updates the estimate $\widehat{w}_t$. 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 $G = (V, E, \widehat{w}_t)$. Once the stopping condition is satisfied, 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 define the design matrix for $p \in \mathcal{P}$ as $\Lambda_p = \sum_{S \in \mathcal{S}} p(S) \boldsymbol{\chi}_{E(S)} \boldsymbol{\chi}_{E(S)}^\top$. We define ρ_{Λ_p} as $\rho_{\Lambda_p} = \max_{x \in [-1,1]^m} \|x\|^2_{\Lambda_p^{-1}}$. Let $\Delta_{\min}$ be the minimal gap between the optimal value and the second optimal value, i.e., $\Delta_{\min} = \min_{S \subseteq V \,:\, S \neq S^*} f_w(S^*) - f_w(S)$. The next theorem shows an upper bound of the number of queries required by Algorithm 2 to output $S_{\mathtt{OUT}} \subseteq V$ that satisfies $\Pr[f_w(S^*) - f_w(S_{\mathtt{OUT}}) \leq \varepsilon] \geq 1 - \delta$.

Theorem 1. *Define $H_\epsilon = \frac{\rho_{\Lambda_p} + \epsilon}{(\Delta_{\min} + \epsilon)^2}$. Then, with probability at least $1-\delta$, DS-Lin (Algorithm 2) outputs $S \subseteq V$ whose density is at least $f_w(S^*) - \epsilon$ and the total number of samples τ is bounded as follows:*

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

$$\tau = O\left(\left(\deg_{\max}^2 R^2 \log \frac{1}{\delta} + \lambda L^2\right) H_\epsilon\right),$$

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 $\epsilon > 0$, parameter $\delta \in (0, 1)$, and allocation strategy p
Output : $S \subseteq V$
for $t = 1, \ldots, m$ **do**
  Choose $S_t \leftarrow \mathrm{argmin}_{S \in \mathrm{supp}(p)} \frac{T_t(S)}{p(S)}$;
  Call the sampling oracle for S_t;
  Observe $r_t(S_t)$;
  $b_{\mathbf{x}_t} \leftarrow b_{\mathbf{x}_{t-1}} + \boldsymbol{\chi}_{E(S_t)} r_t(S_t)$;
end
while *stopping condition* (6) *is not true* **do**
  $t \leftarrow t + 1$;
  Choose $S_t \leftarrow \mathrm{argmin}_{S \in \mathrm{supp}(p)} \frac{T_t(S)}{p(S)}$;
  Call the sampling oracle for S_t and observe $r_t(S_t)$;
  $A_{\mathbf{x}_t} \leftarrow A_{\mathbf{x}_{t-1}} + \boldsymbol{\chi}_{E(S_t)} \boldsymbol{\chi}_{E(S_t)}^\top$;
  $b_{\mathbf{x}_t} \leftarrow b_{\mathbf{x}_{t-1}} + \boldsymbol{\chi}_{E(S_t)} r_{S_t}$;
  $\widehat{w}_t \leftarrow A_{\mathbf{x}_t}^{-1} b_t$;
  If $\widehat{w}_t(e) < 0$ **then** $\widehat{w}_t(e) = 0$ for each $e \in E$;
  $x \leftarrow$ Algorithm 1 for $A_{\mathbf{x}_t}^{-1}$;
  $Z_t \leftarrow C_t \sqrt{\sum_{1 \le i,j \le m} A_{\mathbf{x}_t}^{-1}(i,j) x_i x_j}$;
  $\widehat{S}_t \leftarrow$ Output of the LP-based exact algorithm (Charikar, 2000) for $G(V, E, \widehat{w}_t)$;
end
return $S_{\mathtt{OUT}} \leftarrow \widehat{S}_t$

and if $\lambda \le \frac{\deg_{\max} R^2}{L^2} \log\left(\frac{1}{\delta}\right)$, *then*

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

where $C_{H_\epsilon,\delta}$ *is*

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

The proof of Theorem 2 is given in Appenfix F. Note that $\rho_{\Lambda_p} = d$ holds if we are allowed to query any subset of vertices and employ G-allocation strategy, i.e., $p = \mathrm{argmin}_{p \in \mathcal{P}} \max_{S \subseteq V} \|\boldsymbol{\chi}_{E(S)}\|^2_{\Lambda_p{}^{-1}}$, 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 ρ_{Λ_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 $\Omega(\sum_{e \in [m]} \frac{1}{\Delta_e^2} \log \frac{1}{\delta})$, where m is the number of base arms and Δ_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., $M^* = \mathrm{argmax}_{M \in \mathcal{M}} \sum_{e \in M} w_e$. For each base arm $e \in [m]$, the gap Δ_e is defined as $\Delta_e = \sum_{e \in M^*} w_e - \max_{M \in \mathcal{M} : e \in M} \sum_{e \in M} w_e$ (if $e \notin M^*$), and $\Delta_e = \sum_{e \in M^*} w_e - \max_{M \in \mathcal{M} : e \notin M} \sum_{e \in M} w_e$ (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 $\Omega(\mathbf{H}_{\mathbf{\Lambda}} + \mathbf{H}_{\mathbf{\Lambda}} m^{-1} \log(\delta^{-1}))$, where $\mathbf{H}_{\mathbf{\Lambda}} = \sum_{i=1^m} \frac{1}{\Lambda_i^2}$. In their definition of $\mathbf{H}_{\mathbf{\Lambda}}$, the term Λ_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 $O(H_\epsilon \log \delta^{-1} + H_\epsilon \log(H_\epsilon \log \delta^{-1}))$ if we ignore the terms irrespective of H_ϵ and δ.

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 : $S \subseteq V$
$\tilde{\log}(n-1) \leftarrow \sum_{i=1}^{n-1} \frac{1}{i}$;
$\tilde{T}_0 \leftarrow 0$;
For $T_0(v) \leftarrow 0$ for each $v \in V$;
$S_n \leftarrow V$ and $v_0 \leftarrow \emptyset$;
for $t \leftarrow 1, \ldots, n-1$ **do**
  $\tilde{T}_t \leftarrow \left\lceil \frac{T - \sum_{i=1}^{n+1} i}{\tilde{\log}(n-1)(n-t)} \right\rceil$;
  $T'_t \leftarrow \left\lceil \frac{\tilde{T}_t}{2|S_{n-t+1}|} \right\rceil$ and $\tau_t \leftarrow T'_t - T'_{t-1}$;
  for $v \in S_{n-t+1}$ **do**
    Run Algorithm 4 (sampling procedure);
  end
  $\widehat{f}(S_{n-t+1}) \leftarrow \frac{\frac{1}{2}\sum_{v \in S_{n-t+1}} \widehat{\deg}_{S_{n-t+1}}(v,t)}{|S_{n-t+1}|}$;
  $v_t \leftarrow \mathrm{argmin}_{v \in S_{n-t+1}} \widehat{\deg}_{S_{n-t+1}}(v,t)$;
  $S_{n-t} \leftarrow S_{n-t+1} \setminus \{v_t\}$;
end
return $S_{\mathtt{OUT}} \in \{S_2, \ldots, S_n\}$ *that maximizes* $\widehat{f}(S_i)$

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 $v \notin N_{S_{n-t+2}}(v_{t-1})$ **then**
    Sample $E_{S_{n-t+1}}(v)$ for τ_t times;
    $Y_t \leftarrow T_{E_{S_{n-t+1}}(v)}(t-1)\widehat{\deg}_{S_{n-t+2}}(v,t)$;
    $\widehat{\deg}_{S_{n-t+1}}(v,t) \leftarrow \frac{Y_t + \tau_t \hat{X}_{E_{S_{n-t+1}}(v)}(\tau_t)}{T_{E_{S_{n-t+1}}(v)}(t-1) + \tau_t}$;
    $T_{E_{S_{n-t+1}}(v)}(t) \leftarrow T_{E_{S_{n-t+1}}(v)}(t-1) + \tau_t$;
  end
  else
    Sample $E_{S_{n-t+1}}(v)$ for $\sum_{i=1}^{t} \tau_i$ times;
    $\widehat{\deg}_{S_{n-t+1}}(v,t) \leftarrow \hat{X}_{E_{S_{n-t+1}}}(v)(\sum_{i=1}^{t} \tau_i)$;
    $T_{E_{S_{n-t+1}}(v)}(t) \leftarrow \sum_{i=1}^{t} \tau_i$;
  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 $\hat{X}_F(k) = \frac{1}{k}\sum_{s=1}^{k} X_F(s)$ as the empirical mean of weights of F after k samples. For simplicity, we denote $\widehat{\deg}_{S,v}(t) = \hat{X}_{E_S(v)}\left(T_{E(S)}(t)\right)$.

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 $S_n \leftarrow V$. 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 *empirical quality function* $\widehat{f}(S_{n-t+1})$ and specify one vertex v_t that should be removed. In Algorithm 4, we detail the sampling procedure for obtaining the empirical degree of $v \in S_{n-t+1}$. If v was not a neighbor of v_{t-1} in phase $t-1$, the algorithm samples $E_{S_{n-t+1}}(v)$ for τ_t times, where τ_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 $\sum_{i=1}^{t} \tau_i$ times. Our eliminate scheme removes a vertex v_t that minimizes the empirical degree, i.e., $v_t \in \mathrm{argmin}_{v \in S_{n-t+1}} \widehat{\deg}_{S_{n-t+1}}(v,t)$. Finally, after $n-1$ phases have been done, DS-SR outputs $S_{\mathtt{OUT}} \subseteq V$ that maximizes the empirical quality function, i.e., $S_{\mathtt{OUT}} = \mathrm{argmax}_{S_i \in \{S_2, \ldots, S_n\}} \widehat{f}(S_i)$.

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 $\frac{1}{2}f_w(S^*) - \epsilon$, as shown in the following theorem.

Theorem 2. *Given any $T > m$, and assume that the edge weight distribution ϕ_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 $S_{\mathtt{OUT}} \subseteq V$ such that*

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

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

The proof of Theorem 2 is given in Appenfix H. From the theorem, we see that DS-SR requires a budget of $T = O\left(\frac{n^3 \deg_{\max}}{\epsilon^2} \log\left(\frac{\deg_{\max}}{\epsilon}\right)\right)$ 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 $w(e) = \texttt{rand}(1, 20)$ if $e \in E(S^*)$, and $w(e) = \texttt{rand}(1, 100)$ if $e \in E \setminus E(S^*)$, where $\texttt{rand}(\cdot, \cdot)$ 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 $\eta(e) \sim \mathcal{N}(0, 1)$ 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 : $S \subseteq V$

$w_{\mathrm{avg}} \leftarrow \mathbf{0}$;

$t_{\mathrm{e}} \leftarrow 0$ for $e \in E$;

for $t = 1, 2, , \ldots, T$ **do**

  Choose $S_t \subseteq \mathcal{S}$ uniformly at random;

  Call the sampling oracle for S_t and observe $r_t(S_t)$;

  $t_e \leftarrow t_e + 1$ 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 vector 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 $\lambda = 100$ and $R = 1$. In our theoretical analysis, we provided an upper bound of the number of queries required by DS-Lin for $\epsilon > 0$ 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 briefly 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* $W = \times_{e\in E}[\min\{w(e) - 1, 0\}, w(e) + 1]$, which is detailed in Algorithm 7 in Appendix J. For R-Oracle, we set $\gamma = 0.9$ 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 $i \leftarrow |V|, \ldots, 2$ **do**
  Find $v_i \in \mathrm{argmin}_{v\in S_i} \deg_{S_i}(v)$;
  $S_{i-1} \leftarrow S_i \setminus \{v_i\}$;
end
return $S_i \in \{S_1, \ldots, S_{|V|}\}$ *that maximizes* $f_w(S)$

Results. For DS-SR, in order to make $\tilde{T}_t$ 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-

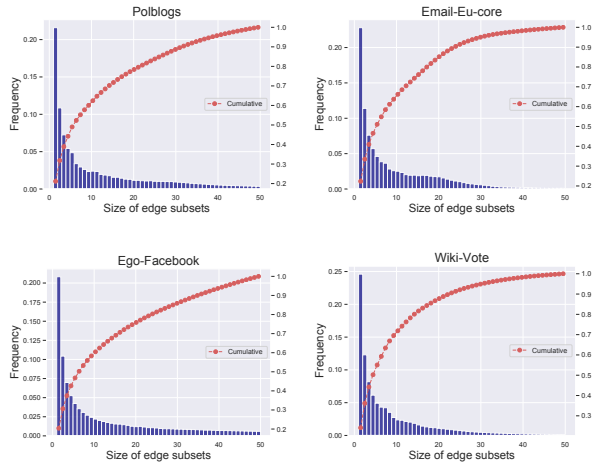

Figure 1. Fraction of the size of edge subsets queried in DS-SR. All values are averaged over 100 executions.

binatorial pure exploration, which we refer to as the DS bandits. We first proposed an (ϵ, δ)-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 $\frac{1}{2}\mathrm{OPT} - \epsilon$. 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 Ré, 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 classification of dense communities in the web. In *Proc. WWW '07*, pp. 461–470, 2007.

Epasto, A., Lattanzi, S., and Sozio, M. Efficient 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 identification: A unified approach to fixed budget and fixed confidence. 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 satisfiability problems using semidefinite 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 efficiently 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 finding 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 identification with full-bandit feedback. *Neural Computation*, 32(9):1733–1773, 2020.

Mahajan, S. and Ramesh, H. Derandomizing approximation algorithms based on semidefinite 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 influence 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. Semidefinite 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 designs. *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.