Robust Densest Subgraph Discovery

Atsushi Miyauchi  
RIKEN AIP  
Tokyo, Japan  
atsushi.miyauchi.hv@riken.jp

Akiko Takeda  
The University of Tokyo / RIKEN AIP  
Tokyo, Japan  
takeda@mist.i.u-tokyo.ac.jp

Abstract—Dense subgraph discovery is an important primitive in graph mining, which has a wide variety of applications in diverse domains. In the densest subgraph problem, given an undirected graph $G = (V, E)$ with an edge-weight vector $w = (w_e)_{e \in E}$, we aim to find $S \subseteq V$ that maximizes the density, i.e., $w(S)/|S|$, where $w(S)$ is the sum of the weights of the edges in the subgraph induced by $S$. Although the densest subgraph problem is one of the most well-studied optimization problems for dense subgraph discovery, there is an implicit strong assumption; it is assumed that the weights of all the edges are known exactly as input. In real-world applications, there are often cases where we have only uncertain information of the edge weights. In this study, we provide a framework for dense subgraph discovery under the uncertainty of edge weights. Specifically, we address such an uncertainty issue using the theory of robust optimization. First, we formulate our fundamental problem, the robust densest subgraph problem, and present a simple algorithm. We then formulate the robust densest subgraph problem with sampling oracle that models dense subgraph discovery using an edge-weight sampling oracle, and present an algorithm with a strong theoretical performance guarantee. Computational experiments using both synthetic graphs and popular real-world graphs demonstrate the effectiveness of our proposed algorithms.

Index Terms—Graph mining, densest subgraph, uncertainty, robust optimization.

I. INTRODUCTION

Dense subgraph discovery, or extracting a dense component in a graph, is an important primitive in graph mining, which has a wide variety of applications in diverse domains. A typical application is the identification of components that have certain special roles or possess important functions in underlying systems represented by graphs. For example, consider the protein–protein interaction graphs, where vertices represent the proteins within a cell and edges (resp. edge weights) represent the interactions (resp. strength of interactions) among the proteins. The dense components in this graph are likely to be the sets of proteins that exhibit identical or similar functions within the cell. As another example, consider the Web graph, where vertices represent web pages and edges represent the hyperlinks among them. The dense components in this graph are generally communities (i.e., the set of web pages addressing identical or similar topics) and occasionally spam link farms, which are effective for improving Web search engines. Other application examples include identifying regulatory motifs in DNA, decision-making for cost-effective marketing strategies, expert team formation, and real-time story identification in micro-blogging streams.

The densest subgraph problem is one of the most well-studied optimization problems for dense subgraph discovery. Let $G = (V, E)$ be an undirected graph with an edge-weight vector $w = (w_e)_{e \in E}$. We denote by $G[S]$ the subgraph induced by $S \subseteq V$, i.e., $G[S] = (S, E(S))$, where $E(S) = \{(u, v) \in E \mid u, v \in S\}$. For an edge-weight vector $w = (w_e)_{e \in E}$, the density of $S \subseteq V$ is defined as $f_w(S) = w(S)/|S|$, where $w(S)$ is the sum of the weights of the edges in $G[S]$, i.e., $w(S) = \sum_{e \in E(S)} w_e$. In the (weighted) densest subgraph problem, given an undirected graph $G = (V, E)$ with an edge-weight vector $w = (w_e)_{e \in E}$, we aim to find $S \subseteq V$ that maximizes the density $f_w(S) = w(S)/|S|$. An optimal solution is called a densest subgraph.

The densest subgraph problem has recently attracted significant interest because it can be solved exactly in polynomial time and with adequate approximation in almost linear time. There are exact algorithms such as Goldberg’s flow-based algorithm and Charikar’s LP-based algorithm. Moreover, Charikar demonstrated that the greedy peeling algorithm designed by Asahiro et al. is a 1/2-approximation algorithm for the problem. This can be implemented to run in $O(m + n \log n)$ time for weighted graphs and $O(m + n)$ time for unweighted graphs, where $n = |V|$ and $m = |E|$.

However, in the densest subgraph problem, there is an implicit strong assumption; it is assumed that the weights of all the edges are known exactly as input. In numerous real-world applications, there are often cases where we have only uncertain information of the edge weights. For example, consider the protein–protein interaction graphs. In the generation process of such graphs, the edge weights representing the strength of the interactions among the proteins are commonly obtained through biological experiments using measuring instruments with some noises. In such a scenario, we have only the estimated values for true edge weights. Therefore, it is challenging to provide a framework for dense subgraph discovery under the uncertainty of edge weights.

A. Our Contribution

In this study, we provide a framework for dense subgraph discovery under the uncertainty of edge weights. Specifically, we address such an uncertainty issue using the theory of robust optimization.

1 A feasible solution is said to be $\alpha$-approximate if its objective value is greater than or equal to the optimal value times $\alpha$. An algorithm is called an $\alpha$-approximation algorithm if it runs in polynomial time and returns an $\alpha$-approximate solution for any instances.
To model the uncertainty of edge weights in real-world applications, we assume that we have only an edge-weight vector \( w = (w_e)_{e \in E} \) that contains the unknown true edge weight; therefore, we conjecture that we can obtain an estimated value of the true edge weight. The edge-weight space can be considered as a product of the confidence intervals of the true edge weights, each of which (i.e., \([l_e, r_e]\) for \(e \in E\)) can be obtained in practice from theoretically guaranteed lower and upper bounds on the true edge weight or repeated sampling of an estimated value of the true edge weight.

The key question is as follows: In this uncertain situation, how can we evaluate the quality of \(S \subseteq V\)? Note here that as we know nothing about \(w^\text{true}\) apart from the fact that \(w^\text{true} \in W\), we cannot directly use the value of \(f_{\text{true}}(S)\) for evaluating \(S\). To answer the question, we use a well-known concept in the theory of robust optimization. In the robust optimization paradigm, the quality of a solution for a robust optimization problem is generally evaluated using a measure called the robust ratio. In our scenario, the robust ratio of \(S \subseteq V\) under edge-weight space \(W\) is defined as the multiplicative gap between the density of \(S\) (i.e., \(w^*(S)\)) and the density of \(S^\star_w\) (i.e., \(w^*(S^\star_w)\)) under the worst-case edge-weight vector \(w^* \in W\), where \(S^\star_w\) is an optimal solution to the densest subgraph problem on \(G\) with \(w^*\). Intuitively, \(S \subseteq V\) with a large robust ratio has a density close to the optimal value even on \(G\) with the edge-weight vector selected adversarially from \(W\). Using the robust ratio, we formulate the robust densest subgraph problem as follows: Given an undirected graph \(G = (V, E)\) with an edge-weight space \(W = \times_{e \in E}[l_e, r_e]\), we aim to find \(S \subseteq V\) that maximizes the robust ratio under \(W\).

For the robust densest subgraph problem, we first provide a strong negative result; specifically, we show that there exist some instances \(G = (V, E)\) with \(W = \times_{e \in E}[l_e, r_e]\) for which any (deterministic) algorithm returns \(S \subseteq V\) that has a robust ratio of \(O(1/n)\). Then, in contrast to this negative result, we present a simple algorithm that utilizes an exact algorithm for the (original) densest subgraph problem. We demonstrate that for any instance that satisfies \(\min_{e \in E} l_e > 0\), our algorithm returns \(S \subseteq V\) that has a robust ratio of at least \(\frac{1}{\min_{e \in E} l_e}\). Moreover, we prove that the lower bound on the robust ratio achieved by our proposed algorithm is the best possible except for the constant factor.

The lower bound on the robust ratio achieved by our algorithm (i.e., \(\frac{1}{\min_{e \in E} l_e}\)) is still small, although it is the best possible except for the constant factor. This negative result was caused by the fact that in the robust densest subgraph problem, we were excessively conservative in evaluating the quality of \(S \subseteq V\), that is, we aimed to find \(S \subseteq V\) that has a relatively large density compared to the optimal value on \(G\) with any edge-weight vector \(w \in W\). In some real-world applications, each confidence interval (i.e., \([l_e, r_e]\) for \(e \in E\)) may be obtained from repeated sampling of an estimated value of the true edge weight; therefore, we conjecture that we can obtain a significantly better lower bound on the robust ratio by using such samplings more sophisticatedly.

To this end, we formulate the robust densest subgraph problem with sampling oracle as follows: We are given an undirected graph \(G = (V, E)\) with an edge-weight space \(W = \times_{e \in E}[l_e, r_e]\), wherein the unknown true edge-weight vector \(w^\text{true} = (w^\text{true}_e)_{e \in E}\) exists. In addition, we have access to an edge-weight sampling oracle that accepts an edge \(e \in E\) as input and returns a real value as output, in time \(\theta\), that was drawn independently from a distribution on \([l_e, r_e]\) in which the expected value is equal to the true edge weight \(w^\text{true}_e\). Given \(\gamma \in (0, 1)\), we aim to find \(S_{\text{out}} \subseteq V\) that satisfies \(w^\text{true} \in W_{\text{out}}\) with a probability of at least \(1 - \gamma\) and \(S_{\text{out}} \subseteq V\) that maximizes the robust ratio under \(W_{\text{out}}\). An important fact is that if we obtain \(S_{\text{out}}\) with an objective function value of \(\alpha\), the subset \(S_{\text{out}}\) is an \(\alpha\)-approximate solution for the densest subgraph problem on \(G\) with \(w^\text{true}\), with a probability of at least \(1 - \gamma\).

For the robust densest subgraph problem with sampling oracle, we present an algorithm with a strong theoretical performance guarantee. Specifically, for any \(\gamma \in (0, 1)\) and \(\epsilon > 0\), our algorithm obtains \(W_{\text{out}} \subseteq W\) that satisfies \(w^\text{true} \in W_{\text{out}}\) with a probability of at least \(1 - \gamma\) and \(S_{\text{out}} \subseteq V\) that has a robust ratio of at least \(1 - \epsilon\) under the edge-weight space \(W_{\text{out}}\), in time pseudo-polynomial in the size of \(G\) and \(W\), \(\theta\), and \(1/\epsilon\). Therefore, we observe that our algorithm obtains a \((1 - \epsilon)\)-approximate solution for the densest subgraph problem on \(G\) with \(w^\text{true}\), with a probability of at least \(1 - \gamma\).

Finally, we conduct computational experiments to evaluate the effectiveness of our proposed algorithms in terms of both the quality of solutions and computation time. We compare our proposed algorithms with a certain baseline algorithm using both synthetic graphs and popular real-world graphs. To generate synthetic graphs appropriate for our experimental evaluation, we introduce a random graph model, which we refer to as the planted uncertain dense subgraph model. With regard to real-world graphs, we introduce a random model for constructing an edge-weight space and a true edge-weight vector for a given graph, which we refer to as the knockout densest subgraph model. The results demonstrate the effectiveness of our proposed algorithms.

B. Related Work

Robust optimization, which has been actively studied in the field of operations research, is known to be an effective methodology for addressing optimization problems under uncertainty [7]–[9]. Recently, the theory of robust optimization has been widely applied to tasks in knowledge discovery and data mining, particularly to graph mining tasks. For example, Chen et al. [14] and He and Kempe [22] studied robust influence maximization, which is a robust variation of the popular graph mining task called influence maximization. Their focus was on the influence maximization counterpart of our work; they aimed to find a subset of vertices that exhibits a large robust ratio in terms of the influence. In particular, Chen et al. [14] developed an algorithm with a theoretical performance guarantee using a certain sampling oracle. To the best of our knowledge, we are the first to utilize the theory of
robust optimization for addressing dense subgraph discovery under uncertainty.

Apart from the uncertainty of edge weights, a large body of work has been devoted to graph mining tasks with the uncertainty of the existence of edges. In this scenario, it is generally assumed that we are given an uncertain graph, i.e., a graph \( G = (V, E) \) with a function \( p : E \rightarrow [0, 1] \) in which \( e \in E \) is present with probability \( p(e) \) whereas \( e \in E \) is absent with probability \( 1 - p(e) \). For a number of fundamental optimization problems on graphs, their counterparts on uncertain graphs have been introduced \([25]\). In particular, Zou \([37]\) studied the densest subgraph problem on uncertain graphs. In this problem, given an uncertain graph \( G = (V, E) \) with a function \( p : E \rightarrow [0, 1] \), we are asked to find \( S \subseteq V \) that maximizes the expected value of the density. Zou \([37]\) demonstrated that this problem can be reduced to the (original weighted) densest subgraph problem and developed a polynomial-time exact algorithm using the reduction. It should be noted that the problems we formulate in the present study cannot be addressed using uncertain graphs. In fact, uncertain graphs do not consider the uncertainty of edge weights; they only model the uncertainty of the existence of edges.

In addition to the variant on uncertain graphs, the densest subgraph problem has numerous noteworthy problem variations. Examples include the size-constraint variants \([1, 10, 17, 27, 34]\) and the variants generalizing the term \( w(S) \) in the density \([30, 32, 35]\) and the term \( |S| \) in the density \([26]\). Furthermore, a large body of work has been devoted to the streaming or dynamic settings of the densest subgraph problem \([5, 11, 16, 24, 29, 33]\). Some literatures have considered the densest subgraph problem on hypergraphs \([24, 31]\) or on multilayer networks \([19]\).

C. Paper Organization

In Section \(\text{II}\) we revisit some existing algorithms for the densest subgraph problem, which will be used in the design of our proposed algorithms. In Section \(\text{III}\) we formulate the robust densest subgraph problem and present a simple algorithm. Then, in Section \(\text{IV}\) we formulate the robust densest subgraph problem with sampling oracle and present an algorithm with a strong theoretical performance guarantee. We report the results of our computational experiments in Section \(\text{V}\). We conclude the study in Section \(\text{VI}\).

II. PRELIMINARIES

Here, we describe Charikar’s LP-based exact algorithm for the densest subgraph problem \([13]\), which will be used in the design of our proposed algorithms. The algorithm introduces a variable \( x_e \) for each \( e \in E \) and a variable \( y_v \) for each \( v \in V \), and solves the following LP in polynomial time:

\[
\begin{align*}
\text{maximize} & \quad \sum_{e \in E} w_e x_e \\
\text{subject to} & \quad x_e \leq y_u, \quad x_e \leq y_v, \quad \forall e = \{u, v\} \in E, \\
& \quad \sum_{v \in V} y_v = 1, \\
& \quad x_e, y_v \geq 0 \quad \forall e \in E, \forall v \in V.
\end{align*}
\]

Intuitively, this LP is a standardized (i.e., linearized) version of a continuous relaxation of the original problem. Let \((x^\ast, y^\ast)\) be an optimal solution to this LP. For a real parameter \( r \geq 0 \), the algorithm introduces a sequence of subsets \( S(r) = \{ v \in V \mid y^\ast_v \geq r \} \) and finds \( r^\ast \in \arg\max_{r \in [0, 1]} f_w(S(r)) \). It should be noted that such \( r^\ast \) can be found by simply examining \( r = y^\ast_v \) for each \( v \in V \). Finally, the algorithm returns \( S(r^\ast) \). Charikar \([13]\) established that the output of the algorithm, i.e., \( S(r^\ast) \), is an optimal solution to the densest subgraph problem.

The above LP-based algorithm is elegant and convenient to implement (if we use a mathematical programming solver such as Gurobi Optimizer or IBM ILOG CPLEX); however, in practice, it is applicable only to graphs with a maximum of hundreds of thousands of edges. Recently, Balalau et al. \([6]\) developed a highly effective preprocessing algorithm for the densest subgraph problem. Their preprocessing algorithm first runs the greedy peeling algorithm to obtain a \( 1/2 \)-approximate solution \( S_{\text{approx}} \subseteq V \). Specifically, the greedy peeling algorithm iteratively removes a vertex with the smallest (weighted) degree in a current remaining graph to obtain a sequence of subsets from \( V \) to \( \emptyset \) and returns the best subset among the sequence. Then, the preprocessing algorithm removes every vertex whose weighted degree is strictly less than \( f_w(S_{\text{approx}}) \). Balalau et al. \([6]\) indicated that this preprocessing does not remove any vertex contained in \( S^\ast \subseteq V \), where \( S^\ast \) is an arbitrary optimal solution to the densest subgraph problem. Therefore, whenever we wish to obtain an optimal solution to the densest subgraph problem, we can apply Balalau et al.’s preprocessing to the input. It should be noted that in practice, Charikar’s LP-based algorithm in combination with Balalau et al.’s preprocessing can obtain an optimal solution in reasonable time (i.e., a few tens of minutes) even on graphs with a few millions of edges.

III. ROBUST DENSEST SUBGRAPH PROBLEM

In this section, we formulate the robust densest subgraph problem and present a simple algorithm.

A. Problem Definition

To model the uncertainty of edge weights in real-world applications, we assume that we have only an edge-weight space \( W = \times_{e \in E} [l_e, r_e] \subseteq \times_{e \in E} [0, \infty) \) (rather than an edge-weight vector \( w = (w_e)_{e \in E} \) that contains the unknown true edge-weight vector \( w^{\text{true}} = (w_e^{\text{true}})_{e \in E} \)). As we know nothing about \( w^{\text{true}} \) except for the fact \( w^{\text{true}} \in W \), we cannot directly use the value of \( f_{w^{\text{true}}}(S) \) for evaluating \( S \subseteq V \). Here, we use
a well-known concept in the theory of robust optimization, which is called the robust ratio. In our scenario, the robust ratio of $S \subseteq V$ under edge-weight space $W$ is defined as

$$\min_{w \in W} \frac{f_w(S)}{f_w(S_w^*)},$$

where $S_w^* \subseteq V$ is a densest subgraph on $G$ with edge-weight vector $w$. Intuitively, $S \subseteq V$ with a large robust ratio has a density close to the optimal value even on $G$ with the edge-weight vector selected adversarially from $W$. Using the robust ratio, we formulate the robust densest subgraph problem as follows:

**Problem 1** (Robust densest subgraph problem). Given an undirected graph $G = (V, E)$ with an edge-weight space $W = \times_{e \in E}[l_e, r_e] \subseteq \times_{e \in E}[0, \infty)$, we are asked to find a subset of vertices $S \subseteq V$ that maximizes the robust ratio under edge-weight space $W$:

$$\min_{w \in W} \frac{f_w(S)}{f_w(S_w^*)},$$

where $S_w^* \subseteq V$ is a densest subgraph on $G$ with edge-weight vector $w$.

This problem is a generalization of the (original) densest subgraph problem. In fact, if $l_e = r_e$ holds for every $e \in E$, the problem reduces to the densest subgraph problem.

Unfortunately, we have the following strong negative result for the robust densest subgraph problem.

**Theorem 1.** There exists an instance of the robust densest subgraph problem (Problem 1) for which any (deterministic) algorithm returns $S \subseteq V$ that has a robust ratio of $O(1/n)$.

**Proof.** Let $G = (V, E)$ be any graph in which every vertex has degree of at least one. We take $W = \times_{e \in E}[l_e, r_e]$ that satisfies $l_e = 0$ and $r_e > 0$ for each $e \in E$. Note that any deterministic algorithm for Problem 1 returns some $S \subseteq V$.

For any $S \subseteq V$, there exists an edge $e' \in E \setminus E(S)$. We can construct an edge weight $w' = (w'_e)_{e \in E}$ such that for each $e \in E$,

$$w'_e = \begin{cases} r_e & \text{if } e = e', \\ 0 & \text{otherwise}. \end{cases}$$

As $w' \in W$ holds, the robust ratio of $S$ can be upper bounded as follows:

$$\min_{w \in W} \frac{f_w(S)}{f_w(S_w^*)} \leq \frac{f_w(S)}{f_w(S_w^*)} = 0.$$

On the other hand, let $S = V$. Let us select an arbitrary edge $e' \in E$. We can again construct an edge weight $w' = (w'_e)_{e \in E}$ such that for each $e \in E$,

$$w'_e = \begin{cases} r_e & \text{if } e = e', \\ 0 & \text{otherwise}. \end{cases}$$

As $w' \in W$ again holds, the robust ratio of $S (= V)$ can be upper bounded as follows:

$$\min_{w \in W} \frac{f_w(S)}{f_w(S_w^*)} \leq \frac{f_w(S)}{f_w(S_w^*)} \leq \frac{r_e}{r_e/2} = O\left(\frac{1}{n}\right).$$

Thus, we have the theorem.

**B. Algorithm and Analysis**

In contrast to the above negative result, we now present a simple algorithm for the robust densest subgraph problem, which utilizes an exact algorithm for the (original) densest subgraph problem. Let $w^- = (l_e)_{e \in E}$ and $w^+ = (r_e)_{e \in E}$. Our algorithm computes $S_{w^-}^* \subseteq V$, i.e., a densest subgraph on $G$ with extreme edge weight $w^- = (l_e)_{e \in E}$ and returns it.

For reference, the procedure is described in Algorithm 1. We can again construct an edge weight $w' \in W$ such that for each $e \in E$,

$$w'_e = \begin{cases} r_e & \text{if } e = e', \\ 0 & \text{otherwise}. \end{cases}$$

As $w' \in W$ again holds, the robust ratio of $S (= V)$ can be upper bounded as follows:

$$\min_{w \in W} \frac{f_w(S)}{f_w(S_w^*)} \leq \frac{f_w(S)}{f_w(S_w^*)} \leq \frac{r_e}{r_e/2} = O\left(\frac{1}{n}\right).$$

Thus, we have the theorem.

**Algorithm 1:** Basic algorithm

**Input:** $G = (V, E)$ with $W = \times_{e \in E}[l_e, r_e]$

**Output:** $S_{\text{out}} \subseteq V$

$S_{\text{out}} \leftarrow$ A densest subgraph on $G$ with edge-weight vector $w^- = (l_e)_{e \in E};$

return $S_{\text{out}}$.

The following theorem provides the theoretical performance guarantee of Algorithm 1. To this end, we use the following lemma, which provides the fundamental property of the density function, i.e., the monotonicity of $f_w(S)$ with respect to edge-weight vector $w$. The proof is straightforward and therefore omitted.

**Lemma 1.** Let $G = (V, E)$ be an undirected graph. Let $w_1$ and $w_2$ be edge-weight vectors such that $w_1 \leq w_2$ holds.

Then, for any $S \subseteq V$, it holds that $f_{w_1}(S) \leq f_{w_2}(S)$.

The following theorem provides the theoretical performance guarantee of Algorithm 1. More specifically, the theorem presents a lower bound on the robust ratio of the output of Algorithm 1 under a certain reasonable condition.

**Theorem 2.** Let $G = (V, E)$ with $W = \times_{e \in E}[l_e, r_e]$ be an instance of the robust densest subgraph problem (Problem 1). Suppose that $\min_{e \in E} l_e > 0$ holds. Then, Algorithm 1 returns $S \subseteq V$ that has a robust ratio of at least $\frac{1}{\max_{e \in E} \frac{l_e}{r_e}}$.

**Proof.** Recall that Algorithm 1 returns $S_{w^-}^*$, i.e., a densest subgraph on $G$ with extreme edge weight $w^- = (l_e)_{e \in E}$.

Then, we can lower bound the robust ratio as follows:

$$\min_{w \in W} \frac{f_w(S_{w^-})}{f_w(S_{w^+})} \geq \frac{f_{w^-}(S_{w^-})}{f_{w^+}(S_{w^+})} \geq \frac{f_{w^-}(S_{w^+})}{f_{w^+}(S_{w^+})} \geq \frac{1}{\max_{e \in E} \frac{l_e}{r_e}} \cdot \frac{f_{w^+}(S_{w^+})}{f_{w^+}(S_{w^+})} = \frac{1}{\max_{e \in E} \frac{l_e}{r_e}},$$

where the first inequality follows from Lemma 1 with the fact that $w^- \leq w \leq w^+$ for any $w \in W$ and the optimality of $S_{w^+}$ in terms of the edge-weight vector $w^+$. 

This lower bound on the robust ratio is significantly better than the upper bound presented in Theorem 1.
bound in Theorem 1 becomes zero as \( n \) increases, whereas \( \max_{e \in E} \log \frac{1}{e} \) does not. However, it should be noted that Theorem 2 does not contradict Theorem 1 because Theorem 2 supposes that \( \min_{e \in E} \log \frac{1}{e} > 0 \) holds.

The following theorem indicates that the lower bound on the robust ratio achieved by Algorithm 1 is the best possible except for the constant factor.

**Theorem 3.** There exists an instance \( G = (V, E) \) with \( W = x \cdot e \in \mathbb{E}[l_e, r_e] \) of the robust densest subgraph problem (Problem 1) that satisfies \( \min_{e \in E} l_e > 0 \) for which any (deterministic) algorithm returns \( S \subseteq V \) that has a robust ratio of \( O\left(\frac{\max_{e \in E} \log e}{\min_{e \in E} t_e}\right) \).

*Proof. Let \( G = (V, E) \) be any graph in which every vertex has degree of at least one and for any \( S \subseteq V \), it holds that \( |E(S)| \leq \alpha |S| \) for some constant \( \alpha \). We take \( W = x \cdot e \in \mathbb{E}[l_e, r_e] \) that satisfies the following three conditions: (i) \( l_e = l \) and \( r_e = r \) for some \( l \) and \( r \), respectively; (ii) \( l > 0 \); and (iii) \( n \geq r/l \). Note that any deterministic algorithm for Problem 1 returns some \( S \subseteq V \).

For any \( S \subseteq V \), there exists an edge \( e' \in E \setminus E(S) \). We can construct an edge-weight vector \( w' = (w'_e)_{e \in E} \) such that for each \( e \in E \),

\[
    w'_e = \begin{cases} 
        r & \text{if } e = e', \\
        t & \text{otherwise}.
    \end{cases}
\]

As \( w' \in W \) holds, the robust ratio of \( S \) can be upper bounded as follows:

\[
    \min_{w \in W} \frac{f_w(S)}{f_w(S_w^*)} \leq \frac{f_{w'}(S)}{f_{w'}(S_{w'}^*)} \leq \frac{\alpha |S| \cdot 1/|S|}{r/2} < 2\alpha \frac{r}{r/l} = O\left(\frac{1}{r/l}\right),
\]

On the other hand, let \( S = V \). Select an arbitrary edge \( e' \in E \). We can again construct an edge weight \( w' = (w'_e)_{e \in E} \) such that for each \( e \in E \),

\[
    w'_e = \begin{cases} 
        r & \text{if } e = e', \\
        t & \text{otherwise}.
    \end{cases}
\]

As \( w' \in W \) again holds, the robust ratio of \( S = V \) can be evaluated as follows:

\[
    \min_{w \in W} \frac{f_w(S)}{f_w(S_w^*)} \leq \frac{f_{w'}(S)}{f_{w'}(S_{w'}^*)} \leq \frac{(\alpha n - 1)l + r}{n} \frac{r/2}{r/2} < \frac{\alpha l + r}{r/2} \leq 2\alpha \frac{r}{r/l} = O\left(\frac{1}{r/l}\right),
\]

where the last inequality follows from the fact that \( n \geq r/l \) holds. Since \( r/l = \max_{e \in E} \frac{\log e}{t_e} \) holds, we have the theorem. \( \square \)

**IV. Robust Densest Subgraph Problem with Sampling Oracle**

In this section, we formulate the robust densest subgraph problem with sampling oracle and present an algorithm with a strong theoretical performance guarantee.

**A. Problem Definition**

The lower bound on the robust ratio achieved by Algorithm 1 (i.e., \( \frac{1}{\min_{e \in E} \log e} \)) is still small, although it is the best possible except for the constant factor. This negative result was caused by the fact that in the robust densest subgraph problem, we were excessively conservative in evaluating the quality of \( S \subseteq V \), that is, we aimed to find \( S \subseteq V \) that has a relatively large density compared to the optimal value on \( G \) with any edge-weight vector \( w \in W \). In some real-world applications, each confidence interval (i.e., \([l_e, r_e]\) for \( e \in E \)) may be obtained from repeated sampling of an estimated value of the true edge weight; therefore, we conjecture that we can obtain a significantly better lower bound on the robust ratio by using such samplings more sophisticatedally.

To this end, we now formulate the robust densest subgraph problem with sampling oracle as follows.

**Problem 2** (Robust densest subgraph problem with sample oracle). We are given an undirected graph \( G = (V, E) \) with an edge-weight space \( W = x \cdot e \in \mathbb{E}[l_e, r_e] \subseteq \mathbb{E}[0, \infty] \), wherein the unknown true edge-weight vector \( w^\text{true} = (w^\text{true}_e)_{e \in E} \) exists. In addition, we have access to an edge-weight sampling oracle that accepts an edge \( e \in E \) as input and returns a real value as output, in time \( \theta \), that was drawn independently from a distribution on \([l_e, r_e]\) in which the expected value is equal to the true edge weight \( w^\text{true}_e \). Given \( \gamma \in (0, 1) \), we are asked to find

- \( W_{\text{out}} \subseteq W \) that satisfies \( \Pr[w^\text{true} \in W_{\text{out}}] \geq 1 - \gamma \)
- \( S_{\text{out}} \subseteq V \) that maximizes the robust ratio under edge-weight space \( W_{\text{out}} \), i.e.,

\[
    \min_{w \in W_{\text{out}}} \frac{f_w(S_{\text{out}})}{f_w(S_{w_{\text{out}}}^*)},
\]

where \( S_{w_{\text{out}}}^* \subseteq V \) is a densest subgraph on \( G \) with edge-weight vector \( w \).

Let \((W_{\text{out}}, S_{\text{out}})\) be an output of Problem 2. Since \( w^\text{true} \in W_{\text{out}} \) holds with a probability of at least \( 1 - \gamma \), the following inequality

\[
    \frac{f_{w^\text{true}}(S_{\text{out}})}{f_{w^\text{true}}(S_{w^\text{true}}^*)} \geq \min_{w \in W_{\text{out}}} \frac{f_w(S_{\text{out}})}{f_w(S_{w_{\text{out}}}^*)}
\]

also holds with a probability of at least \( 1 - \gamma \). Therefore, if \( S_{\text{out}} \) has an objective function value of \( \alpha \), we observe that \( S_{\text{out}} \) is an \( \alpha \)-approximate solution for the densest subgraph problem on \( G \) with \( w^\text{true} \), with a probability of at least \( 1 - \gamma \).

**B. Algorithm and Analysis**

Here, we present an algorithm for Problem 2 with a strong theoretical performance guarantee. Our algorithm first obtains \( S_{w_{\text{out}}}^* \), i.e., a densest subgraph on \( G \) with extreme edge weight \( w^- = (l_e)_{e \in E} \), to compute the value of \( f_{w^-}(S_{w^-}^*) \). Then, for each \( e \in E \), the algorithm iteratively obtains estimated values for the true edge weight of \( e \) using a sampling oracle for an appropriate number of times, say \( t_e \), which will be defined later. Note that \( t_e \) is determined using the value of \( f_{w^-}(S_{w^-}^*) \). Using the estimated values, the algorithm constructs an edge-weight
Algorithm 2: Algorithm with a sampling oracle

Input: \( G = (V, E) \) with \( W = x_{e \in E} [l_e, r_e] \) (satisfying \( \max_{e \in E} l_e > 0 \)), a sampling oracle, \( \gamma \in (0, 1) \), and \( \epsilon > 0 \)

Output: \((W_{\text{out}}, S_{\text{out}})\) such that \( W_{\text{out}} \subseteq W \) and \( S_{\text{out}} \subseteq V \)

\( S'_{w^-} \leftarrow \) A densest subgraph on \( G \) with \( w^- = (l_e)_{e \in E} \);
for each \( e \in E \) do
  if \( l_e = r_e \) then
    \( \hat{r}_e \leftarrow l_e, \hat{r}_{\text{out}} \leftarrow r_e \);
  else
    \( t_e \leftarrow \left[ \frac{m(r_e - l_e)^2 \ln(2m)}{\epsilon^2 f_w(S'_{w^-})^2} \right] ; \)
    Call the sampling oracle for \( e \) for \( t_e \) times and observe \( w_1^e, \ldots, w_{t_e}^e \);
    \( \hat{p}_e \leftarrow \frac{1}{t_e} \sum_{i=1}^{t_e} w_i^e ; \)
    \( \delta \leftarrow \epsilon f_w(S'_{w^-}) / \sqrt{2m} ; \)
    \( \hat{r}_e \leftarrow \max\{ l_e, \hat{p}_e - \delta \}, r_e \leftarrow \min\{ r_e, \hat{p}_e + \delta \} ; \)
  \( W_{\text{out}} \leftarrow \times_{e \in E} [\hat{r}_e, \hat{r}_{\text{out}}] ; \)
  \( S_{\text{out}} \leftarrow \) A densest subgraph on \( G \) with \( w^- = (\hat{r}_e)_{e \in E} ; \)
end for

return \((W_{\text{out}}, S_{\text{out}})\);

The following lemma is a key ingredient for establishing our theorem, which provides an upper bound on the difference between two density values for \( S \subseteq V \); one with the edge-weight vector \( w_1 \) and the other with the edge-weight vector \( w_2 \), using the distance between the two vectors \( w_1 \) and \( w_2 \).

Lemma 2. Let \( G = (V, E) \) be an undirected graph. Let \( w_1 \) and \( w_2 \) be edge-weight vectors such that \( \|w_1 - w_2\|_\infty \leq \beta \) holds. Then, for any \( S \subseteq V \), it holds that

\[
|f_{w_1}(S) - f_{w_2}(S)| \leq \sqrt{\frac{m}{2}} \cdot \beta.
\]

Proof. We first consider the case where \( |S| < \sqrt{2m} + 1 \) holds. We have

\[
|f_{w_1}(S) - f_{w_2}(S)| = \frac{|w_1(S) - w_2(S)|}{|S|} \leq \frac{(|S| - 1) \beta}{|S|} \leq \sqrt{\frac{m}{2}} \cdot \beta.
\]

Next, we consider the case where \( |S| \geq \sqrt{2m} + 1 \) holds. Since \( |E(S)| \leq m \) holds, we have

\[
|f_{w_1}(S) - f_{w_2}(S)| = \frac{|w_1(S) - w_2(S)|}{|S|} \leq \frac{|E(S)| \cdot \beta}{|S|} \leq \frac{m \cdot \beta}{|S|} \leq \sqrt{\frac{m}{2}} \cdot \beta.
\]

Thus, we have the lemma.

It should be noted that under an assumption identical to that in Lemma 2, we can also obtain an upper bound depending on the size of \( S \subseteq V \), i.e., \( |f_{w_1}(S) - f_{w_2}(S)| \leq \frac{(|S| - 1) \beta}{2} \), which is more effective than the upper bound presented in Lemma 2 in the case where \( |S| < \sqrt{2m} + 1 \) holds. However, when we do not know the size of \( S \) (i.e., we have only \( |S| \leq n \)), the upper bound presented in Lemma 2 is significantly more effective in practice because most real-world graphs are sparse, i.e., \( m = O(n) \) holds. Note that such upper bounds affect the definition of \( t_e \) in Algorithm 2. If we use the above upper bound depending on the size of \( S \) alternatively, we have \( t_e = \frac{(n-1)(r_e - l_e)^2 \ln(2m)}{2 \epsilon^2 f_w(S'_{w^-})^2} \), which is significantly less effective than ours in practice.

In the proof of the above theorem, we use the following form of Hoeffding bound:

Fact 1 (Hoeffding bound; Theorem 2 of Hoeffding [23]). Let \( X_1, \ldots, X_t \) be independent random variables such that \( X_i \in [a_i, b_i] \) holds for any \( i = 1, \ldots, t \). Then, for any \( c > 0 \), it holds that

\[
\Pr \left[ \left| \frac{1}{t} \sum_{i=1}^{t} X_i - \mathbb{E} \left[ \frac{1}{t} \sum_{i=1}^{t} X_i \right] \right| \geq c \right] \leq 2 \exp \left( \frac{-2c^2}{\sum_{i=1}^{t} (b_i - a_i)^2} \right).
\]
\(w_e^{\text{true}} = l_e (= r_e).\) On the other hand, for any \(e \in E\) with \(l_e < r_e\), we have
\[
\Pr \left[ w_e^{\text{true}} \notin \{l_e, r_e \} \right] = \Pr \left[ w_e^{\text{true}} \notin \left[ \min \{l_e, r_e - \delta\}, \min \{r_e, r_e + \delta\} \right] \right] = \Pr \left[ \frac{1}{t_e} \sum_{i=1}^{t_e} w_e^i - w_e^{\text{true}} > \epsilon \cdot f_{w^-}(S_w^*) \right] \leq 2 \exp \left( - \ln \left( \frac{2m}{\gamma^2} \right) \right) \leq 2 \exp \left( - \ln \left( \frac{2m}{\gamma} \right) \right) = \frac{\gamma}{m}.
\]
The second equality follows from the definition of \(\delta\) in the algorithm and the fact that \(w_e^{\text{true}} \in \{l_e, r_e\}.\) The first inequality follows from Fact 1 and the second inequality follows from the definition of \(t_e\) in the algorithm. By a union bound, we have
\[
\Pr \left[ \exists e \in E, \ w_e^{\text{true}} \notin \{l_e, r_e \} \right] \leq \frac{\gamma}{m} \cdot m = \gamma,
\]
which guarantees that \(\Pr[w_e^{\text{true}} \in W_{out}] \geq 1 - \gamma\) holds.

Next, we establish that the output \(S_{out} \subseteq V\) of Algorithm 2 has a robust ratio of at least \(1 - \epsilon\) under edge-weight space \(W_{out}.\) Recall that \(w_{\text{out}}^+ = (l_e)_{e \in E}\) and \(w_{\text{out}}^- = (\epsilon)_{e \in E}.\) Noticing that \(\|w_{\text{out}}^+ - w_{\text{out}}^-\|_{\infty} \leq 2\epsilon = \sqrt{\frac{2}{m}} \cdot \epsilon \cdot f_{w^-}(S_w^*),\) we have
\[
\frac{f_{w_{\text{out}}^+}(S_w^*)}{f_{w_{\text{out}}^-}(S_w^*)} = 1 - \frac{f_{w_{\text{out}}^-}(S_w^*) - f_{w_{\text{out}}^+}(S_w^*)}{f_{w_{\text{out}}^-}(S_w^*)} \geq 1 - \frac{\sqrt{\frac{2}{m}} \cdot \epsilon \cdot f_{w^-}(S_w^*)}{f_{w_{\text{out}}^-}(S_w^*)} \geq 1 - \epsilon,
\]
where the first inequality follows from Lemma 2 with the fact that \(\|w_{\text{out}}^+ - w_{\text{out}}^-\|_{\infty} \leq \sqrt{\frac{2}{m}} \cdot \epsilon \cdot f_{w^-}(S_w^*),\) and the second inequality follows from the optimality of \(S_{w_{\text{out}}^+}\) in terms of the edge-weight vector \(w_{\text{out}}^+ + \text{Lemma 1}\) with the fact that \(w_{\text{out}}^+ \leq w_{\text{out}}^-\).

The output \(S_{out} \subseteq V\) of Algorithm 2 is actually \(S_{w_{\text{out}}^+} \subseteq V.\) Using the above inequality, we can evaluate the robust ratio of \(S_{out}\) under edge-weight space \(W_{out}\) as follows:
\[
\min_{w \in W_{out}} \frac{f_w(S_w^*)}{f_{w_{\text{out}}}(S_{w_{\text{out}}^+})} \geq \frac{f_{w_{\text{out}}^+}(S_w^*)}{f_{w_{\text{out}}^-}(S_{w_{\text{out}}^+})} \geq \frac{f_{w_{\text{out}}^-}(S_w^*)}{f_{w_{\text{out}}^-}(S_{w_{\text{out}}^+})} \geq 1 - \epsilon.
\]
Finally, it is evident that Algorithm 2 runs in time pseudo-polynomial in the size of \(G\) and \(W, \theta,\) and \(1/\epsilon.\)

V. EXPERIMENTAL EVALUATION

The purpose of our experiments is to evaluate the effectiveness of our proposed algorithms (i.e., Algorithms 1 and 2) in terms of the quality of solutions and computation time. To this end, we compare our algorithms with a certain baseline algorithm using both synthetic graphs and popular real-world graphs. The baseline algorithm, denoted by Random, first selects \(w_{\text{rand}}\) from \(W\) uniformly at random; then, it returns a densest subgraph on \(G\) with edge weight \(w_{\text{rand}}.\)

All the algorithms we compare need to compute a densest subgraph on \(G\) with some edge weight \(w.\) To this end, we employed Charikar’s LP-based algorithm in conjunction with Balalau et al.’s preprocessing, which was described in Section 4. To solve the LP relaxations, we used a state-of-the-art mathematical programming solver, Gurobi Optimizer 7.5.1, with default parameter settings except for Method = 1; it stipulates that the LP relaxations are solved using a dual simplex algorithm.

The experiments were conducted on a Linux machine with Intel Xeon Processor E5-2690 v4 2.6 GHz CPU and 256 GB RAM. The code was written in Python, which is publicly available\(^2\).

A. Synthetic Graphs

Here, we report the results of the computational experiments with synthetic graphs. To generate synthetic graphs appropriate for our experimental evaluation, we introduce a random graph model, which we refer to as the planted uncertain dense subgraph model.

In this model, we first generate an Erdős–Rényi random graph \(G = (V, E)\) with \(n\) vertices and edge probability \(p.\) Then, we focus on a subset of vertices \(S' \subseteq V\) consisting of \(n' (\leq n)\) vertices as a planted dense region. On this graph \(G = (V, E),\) we make an edge-weight space \(W = \times_{e \in E} [l_e, r_e]\) as follows: Let \(\alpha \in [0, 0.9]\) be a real parameter. For each \(e \in E,\) we set
\[
[l_e, r_e] = \begin{cases} [\operatorname{rand}(0.1 + \alpha, 1.0), 1.0] & \text{if } e \in E(S'), \\ [0.1, \operatorname{rand}(0.1, 1.0 - \alpha)] & \text{if } e \in E \setminus E(S'). \end{cases}
\]
where \(\operatorname{rand}(\cdot, \cdot)\) is a value selected uniformly at random from the closed interval between the two values within the parenthesis. Note that the larger the parameter \(\alpha,\) the more significant the difference between \(l_e, r_e\) for \(e \in E(S')\) and \(l_e, r_e\) for \(e \in E \setminus E(S').\) For example, when \(\alpha = 0.0,\) each \(e \in E(S')\) has \([\operatorname{rand}(0.1, 1.0), 1.0]\) and each \(e \in E \setminus E(S')\) has \([0.1, \operatorname{rand}(0.1, 1.0)];\) however, when \(\alpha = 0.9,\) each \(e \in E(S')\) has \([l_e, r_e] = [1.0, 1.0]\) and each \(e \in E \setminus E(S')\) has \([l_e, r_e] = [0.1, 0.1].\) Furthermore, we define a true edge-weight vector \(w_{\text{true}} = (w_e)_{e \in E}\) as follows: For each \(e \in E,\) we set
\[
w_e^{\text{true}} = \begin{cases} \operatorname{rand}(\max\{l_e, 0.9\}, 1.0) & \text{if } e \in E(S'), \\ \operatorname{rand}(0.1, \min\{r_e, 0.2\}) & \text{if } e \in E \setminus E(S'). \end{cases}
\]
More or less, \(w_e^{\text{true}}\) tends to exhibit a relatively large value for \(e \in E(S')\) and a relatively small value for \(e \in E \setminus E(S').\) Note that \(w_{\text{true}} \in W\) holds.

Algorithm 2 requires a sampling oracle, which we simulate as follows: For each \(e \in E,\) the sampling oracle returns \(\operatorname{rand}(w_e^{\text{true}} - l_e, w_e^{\text{true}} + r_e),\) where \(\min_e = \min\{w_e^{\text{true}} - l_e, r_e - w_e^{\text{true}}\}.\) It should be noted that for every \(e \in E,\)
\(^2\)https://github.com/atsushi-miyauchi/robust-densest-subgraph-discovery
the expected value is equal to the true edge weight \( w_{\text{true}} \), as required.

Throughout our experiments, we set \( n = 500 \) and \( p = 0.01 \). In these parameter settings, we construct four types of instances with \( n' = 50, 100, 150, \) and \( 200 \); in each of these, the parameter \( \alpha \) varies from \( 0.0 \) to \( 0.9 \) with increments of \( 0.1 \).

The results are shown in Figure 1. The quality of output \( S \subseteq V \) is evaluated by the robust ratio at \( w_{\text{true}} \), i.e., \( f_{w_{\text{true}}}(S)/f_{w_{\text{true}}}(S_{\gamma, \epsilon, \text{true}}) \). With regard to the parameters in Algorithm 2, we set \( (\gamma, \epsilon) = (0.1, 0.5) \). Because Random and Algorithm 2 contain randomness, we performed them 10 times for each graph realization and considered the average value of the robust ratio at \( w_{\text{true}} \) as the result for the graph.

As is evident, our proposed algorithms, Algorithms 1 and 2, outperform the baseline algorithm Random. In particular, owing to the power of the use of a sampling oracle, Algorithm 2 obtains \( S \subseteq V \) with a significantly high robust ratio; the robust ratio almost always attains its upper bound (i.e., 1.0), which implies that the output of Algorithm 2 is (almost always) a densest subgraph on \( G \) with \( w_{\text{true}} \). Note that such a high performance of Algorithm 2 is not a trivial outcome because we set \( (\gamma, \epsilon) = (0.1, 0.5) \). Algorithm 1 outperforms Random, particularly under relatively challenging instances with small \( \alpha \), although it exhibits inferior performance for relatively easy instances with large \( \alpha \). Both Algorithm 1 and Random have higher performances for instances with larger \( n' \); this appears to be a result of the fact that a planted dense region becomes significant as \( n' \) increases.

**B. Real-World Graphs**

Here, we report the results of the computational experiments with real-world graphs. Table 1 lists the real-world graphs on which our computational experiments were conducted; most of these are available in Leskovec and Krevl [28]. As is evident, every graph is sparse, i.e., the average degree \( 2m/n \) is small. Note that all the graphs here were made simple and undirected (if necessary) by omitting the directions of the edges and by removing self-loops and redundant multiple edges. Furthermore, if a graph is not connected, we take only the largest connected component in the graph. To effectively evaluate the robustness of the algorithms, we introduce a random model for constructing an edge-weight space and a true edge-weight vector on a (real-world) graph; we call this model the *knockout densest subgraph model*.

First, we explain the intuition behind the model. Let \( G = (V, E) \) be a given (real-world) undirected graph and \( S^* \subseteq V \) be a densest subgraph on \( G \) (with unweighted edges). Suppose here that we put a very small true edge weight \( t_e \) for each \( e \in E(S^*) \), whereas we put a relatively large true edge weight \( t_e \) for each \( e \in E \setminus E(S^*) \). Suppose also that the edge-weight space only marginally reflects the values of the true edge weights. In such a situation, from the structure (i.e., the existence/non-existence of edges) of the graph, any algorithm that does not consider the edge-weight space or sampling oracle with adequate caution tends to detect \( S^* \) despite the fact that \( S^* \) is no longer likely to be a densest subgraph on \( G \) with \( w_{\text{true}} \).

The knockout densest subgraph model is a random model that simulates the above situation. Specifically, we make an edge-weight space \( W = \times_{e \in E}[l_e, r_e] \) as follows: For each \( e \in E \), we set

\[
[l_e, r_e] = \begin{cases} 
  [0.1, \text{rand}(0.1, 0.9)] & \text{if } e \in E(S^*) , \\
  [\text{rand}(0.2, 1.0), 1.0] & \text{if } e \in E \setminus E(S^*) .
\end{cases}
\]

In addition, we define a true edge-weight vector \( w_{\text{true}} = (w_{\text{true}}(e))_{e \in E} \) as follows: For each \( e \in E \), we set

\[
w_{\text{true}} = \begin{cases} 
  \text{rand}(0.1, \min\{r_e, 0.11\}) & \text{if } e \in E(S^*) , \\
  \text{rand}(\max\{l_e, 0.99\}, 1.0) & \text{if } e \in E \setminus E(S^*) .
\end{cases}
\]

Note that \( w_{\text{true}} \in W \) holds. Algorithm 2 requires a sampling oracle, which we simulate in a manner identical to that in the planted uncertain dense subgraph model.

The results are summarized in Table II. The quality of output \( S \subseteq V \) is again evaluated by the robust ratio at \( w_{\text{true}} \). To observe the scalability, we list the computation time for the algorithms. With regard to Algorithm 2, we also list the average number of calls of the sampling oracle per edge. With regard to the parameters in Algorithm 2, we apply the algorithm to large graphs, we set \( (\gamma, \epsilon) = (0.9, 0.9) \). Moreover, we perform a simple preprocessing algorithm, which was inspired by Balalau et al.’s preprocessing technique, to reduce the size of a given graph. This preprocessing does not impair the theoretical performance guarantee of our algorithm. Owing to space limitations, we omit the details here. With regard to Random and Algorithm 2 we performed them 10 times on each graph and considered the average value of each of the robust ratio at \( w_{\text{true}} \) and the computation time as the results for the graph.
### Table I

Real-world graphs used in our experiments. Here, $|S^*|$ denotes the size of a densest subgraph on a graph (with unweighted edges).

| Name         | $n$  | $m$  | $\frac{2m}{n}$ | $|S^*|$ | Description                      |
|--------------|------|------|----------------|--------|----------------------------------|
| Karate       | 34   | 78   | 4.59           | 16     | Social network                   |
| Lesmis       | 77   | 254  | 6.60           | 23     | Co-appearance network            |
| Polbooks     | 105  | 441  | 8.40           | 24     | Co-purchased network             |
| Adjnoun      | 112  | 425  | 7.59           | 48     | Word adjacency                   |
| Football     | 115  | 613  | 10.66          | 115    | Sports game network              |
| Jazz         | 198  | 2,742| 7.70           | 114    | Social network                   |
| email-Eu-core| 986  | 16,064| 32.58         | 224    | Email communication              |
| Email        | 1,133| 5,451| 9.62           | 301    | Email communication              |
| Polblogs     | 1,222| 16,714| 27.36         | 139    | Blog network                     |
| Wiki-Vote    | 7,866| 425  | 28.51          | 835    | Wikipedia “who-votes-whom”       |
| ca-HepTh     | 6,838| 24,806| 5.74          | 32     | Co-authorship network            |
| ca-HepPh     | 11,204| 117,619| 21.00       | 239    | Co-authorship network            |
| ca-CondMat   | 21,363| 91,286| 8.55          | 30     | Co-authorship network            |
| AS-22july06  | 22,963| 48,436| 10.73        | 555    | Internet at autonomous system level |
| email-Enron  | 33,696| 180,811| 10.73     | 555    | Email communication              |
| web-Stanford | 255,265| 1,941,926| 15.21   | 597    | Web graph                        |
| web-NotreDame| 325,729| 1,090,108| 6.69      | 1,367  | Web graph                        |

### Table II

Results for real-world graphs.

| Name          | Random       | Algorithm 1 | Algorithm 2 |
|---------------|--------------|-------------|-------------|
|               | Ratio | Time(s) | Ratio | Time(s) | Ratio | Time(s) | Calls (avg.) |
| Karate        | 0.992 | 0.00   | 1.00  | 0.00   | 1.00  | 0.01   | 92.65        |
| Lesmis        | 0.992 | 0.01   | 1.00  | 0.01   | 1.00  | 0.03   | 77.12        |
| Polbooks      | 0.992 | 0.02   | 0.98  | 0.01   | 1.00  | 0.06   | 99.97        |
| Adjnoun       | 0.990 | 0.02   | 0.95  | 0.02   | 1.00  | 0.15   | 340.66       |
| Football      | 0.995 | 0.04   | 1.00  | 0.04   | 1.00  | 2.05   | 3991.49      |
| Jazz          | 0.999 | 0.15   | 0.99  | 0.08   | 1.00  | 0.38   | 118.86       |
| emailing-Eu-core| 0.960 | 3.02   | 0.99  | 2.35   | 1.00  | 11.48  | 492.59       |
| Email         | 0.851 | 0.82   | 0.98  | 0.69   | 1.00  | 8.88   | 1918.53      |
| Polblogs      | 0.999 | 1.89   | 0.99  | 1.46   | 1.00  | 5.33   | 186.84       |
| Wiki-Vote     | 0.961 | 50.45  | 0.99  | 54.27  | 1.00  | 201.25 | 1359.74      |
| ca-HepTh      | 1.000 | 1.29   | 1.00  | 1.34   | 1.00  | 7.86   | 677.20       |
| ca-HepPh      | 0.713 | 32.85  | 0.99  | 19.70  | 1.00  | 57.05  | 339.81       |
| ca-CondMat    | 0.999 | 9.03   | 0.99  | 8.76   | 1.00  | 142.98 | 2751.65      |
| AS-22july06   | 0.940 | 2.02   | 0.98  | 2.25   | 1.00  | 8.40   | 623.54       |
| email-Enron   | 0.952 | 106.20 | 0.99  | 179.63 | 1.00  | 374.16 | 1613.85      |
| web-Stanford  | 0.998 | 55.27  | 0.99  | 47.19  | 1.00  | 419.49 | 1129.58      |
| web-NotreDame | 1.000 | 482.26 | 0.99  | 125.07 | 1.00  | 760.15 | 2378.03      |

As is evident, the trend is consistent with the results of the experiments with synthetic graphs; that is, Algorithms 1 and 2 outperform Random. Algorithm 2 (almost always) obtains a densest subgraph on $G$ with $w_{\text{true}}$. Algorithm 1 outperforms Random, particularly in relatively challenging instances for which Random only obtains $S \subseteq V$ with a robust ratio of at most 0.95. Algorithm 2 is not significantly worse in terms of the scalability.

### VI. Conclusion

In this study, we have provided a framework for dense subgraph discovery under the uncertainty of edge weights. Specifically, we have addressed such an uncertainty issue using the theory of robust optimization. First, we formulated the robust densest subgraph problem (Problem 1) and presented a simple algorithm (Algorithm 1). We then formulated the robust densest subgraph problem with sampling oracle (Problem 2) that models dense subgraph discovery using an edge-weight sampling oracle, and presented an algorithm with a strong theoretical performance guarantee (Algorithm 2). Computational experiments using both synthetic graphs and popular real-world graphs demonstrated the effectiveness of our proposed algorithms.

**Acknowledgment**

The authors wish to thank the anonymous reviewers for their valuable comments. The authors also wish to thank Yuko Kuroki for her helpful comments, which improved the presentation of the paper. This work was supported by JST CREST Grant Numbers JPMJCR14D2 and JPMJCR15K5, Japan. A.M. is supported by a Grant-in-Aid for Research Activity Start-up (No. 17H07357).
