Novel Dense Subgraph Discovery Primitives: Risk Aversion and Exclusion Queries

Charalampos E. Tsourakakis[1] (✉), Tianyi Chen[1], Naonori Kakimura[2], and Jakub Pachocki[3]

[1] Boston University, Boston MA, USA
[2] Keio University, Japan
[3] OpenAI, USA

Abstract. In the densest subgraph problem, given an undirected graph $G(V, E, w)$ with non-negative edge weights we are asked to find a set of nodes $S \subseteq V$ that maximizes the degree density $w(S)/|S|$, where $w(S)$ is the sum of the weights of the edges in the graph induced by S. This problem is solvable in polynomial time, and in general is well studied. But what happens when the edge weights are negative? Is the problem still solvable in polynomial time? Also, why should we care about the densest subgraph problem in the presence of negative weights?
In this work we answer the aforementioned questions. Specifically, we provide two novel graph mining primitives that are applicable to a wide variety of applications. Our primitives can be used to answer questions such as "how can we find a dense subgraph in Twitter with lots of replies and mentions but no follows?", "how do we extract a dense subgraph with high expected reward and low risk from an uncertain graph"? We formulate both problems mathematically as special instances of dense subgraph discovery in graphs with negative weights. We study the hardness of the problem, and we prove that the problem in general is **NP**-hard, but we also provide sufficient conditions under which it is poly-time solvable. We design an efficient approximation algorithm that works best in the presence of small negative weights, and an effective heuristic for the more general case. Finally, we perform experiments on various real-world datasets that verify the value of the proposed primitives, and the effectiveness of our proposed algorithms.
The code and the data are available at `https://github.com/negativedsd`.

1 Introduction

Dense subgraph discovery (abbreviated as *DSD* henceforth) is a major and active topic of research in the fields of graph algorithms and graph mining. A wide range of real-world, data mining applications rely on DSD including correlation mining, fraud detection, electronic commerce, bioinformatics, mining Twitter data, efficient algorithm design for fast distance queries in massive networks, and graph compression [15].

In this work we introduce two novel primitives for DSD. These two primitives are strongly motivated by real-world applications that we discuss in greater

detail in Section 3.1. The first question that our work addresses is related to uncertain graphs. Uncertain graphs appear in a wide variety of applications that we survey in Section 2. We define the uncertain graph model we use formally in Section 3.1, but intuitively, uncertain graphs model probabilistically real-world scenarios where each edge may exist or not in a graph (e.g., failure of a link). Problem 1 aims to find a *risk-averse* dense subgraph. A similar formulation was suggested recently by Tsourakakis et al. for graph matchings [31].

Problem 1 (Risk-averse DSD). Given an uncertain graph $\mathcal{G}$, how do we find a set of nodes S that induces a dense subgraph in expectation, and the probability of not being dense in a realization/sample of $\mathcal{G}$ is low?

Our second problem focuses on multigraphs whose edges are associated with different types. Such graphs appear naturally in numerous applications, and are also known as multilayer multigraphs, e.g., [11]. For example, similarity between two videos can be defined based on different criteria, e.g., audio, visual, and how frequently these videos are being co-watched on Youtube. Similarity between time series can be defined using a variety of measures including Euclidean distance, Fourier coefficients, dynamic time wraping, edit distance among others [28]. Emails between people can be classified based on the nature of the interaction (e.g., business, family). Twitter users may interact in various ways, including *follow, reply, mention, retweet, like*, and *quote*. We formulate Problem 2, whose goal is to detect efficiently dense subgraphs that exclude certain types of edges. Later, we will define two variations, soft- and hard-exclusion queries.

Problem 2 (DSD-Exclusion-Queries). Given a multigraph $G(V, E, \ell)$, where $\ell : E \to \{1, \ldots, L\} = [L]$ is the labeling function, and L is the number of types of interactions, and an input set $\mathcal{I} \subseteq [L]$ of interactions, how do we find a set of nodes S that induces a dense subgraph but does not induce any edge e such that $\ell(e) \in \mathcal{I}$?

Contributions. Our contributions are summarized as follows.

- We introduce two novel problems, (i) risk averse DSD, and (ii) DSD in large-scale multilayer networks with exclusion queries. We show in Section 3.1 that these two problems are special cases of DSD in undirected graphs with negative weights. To the best of our knowledge, this is the first work that introduces these algorithmic primitives.
- We prove that DSD in the presence of negative weights is **NP**-hard in general by reducing Max-Cut to our problem (Section 3.2), but we also provide sufficient conditions under which it is exactly solvable in polynomial time.
- We design a space-, and time- efficient approximation algorithm that performs best in the presence of small negative weights. In the case of existence of large negative weights, we design a well-performing heuristic.

- We deploy our developed primitives on the two real-world applications we introduce. We extract subgraphs from uncertain graphs with high expected induced weight and low risk. Finally, we mine Twitter data by finding dense subgraphs that exclude certain types of interactions. A non-trivial experimental contribution is the creation of an uncertain graph from the TMDB database and Twitter graphs from the Greek Twitter-verse, that we will make available to the community. Our tools provide insights, and we are confident that will find numerous applications in graph mining, and anomaly detection.

In the following sections we use the notation summarized in Table 1.

Table 1. Notation

Notation	Description
$deg^+(u)$ $(deg^-(u))$	Positive (negative) degree of node u. $deg^+(u) > 0$, $deg^-(u) > 0$
$d(u)$	Total degree of u. $d(u) = deg^+(u) - deg^-(u)$
$w^+(e)(w^-(e))$	Positive (negative) weight of edge e. $w^+(e) > 0$, $w^-(e) > 0$
$deg_S^+(u)$ $(deg_S^-(u))$	Positive (negative) degree of node u within node set $S \subseteq V$. $deg_S^+(u) = \sum_{v \in S} w^+(u,v)$, $deg_S^-(u) = \sum_{v \in S} w^-(u,v)$
$w^+(S)$ $(w^-(S))$	Total positive (negative) induced weight by S. $w^+(S) = \sum_{e \in E(S)} w^+(e)$, $w^-(S) = \sum_{e \in E(S)} w^-(e)$
$d_S(u)$	Total degree of node u within S. $d_S(u) = deg_S^+(u) - deg_S^-(u)$

2 Related Work

Uncertain graphs model naturally a wide variety of datasets and applications including protein-protein interactions [21], kidney exchanges [27], and influence maximization [19] While a lot of research work has focused on designing graph mining algorithms for uncertain graphs, e.g., [6,22,24], there is less work on designing efficient risk-averse optimization algorithms, and even lesser with solid theoretical guarantees.

Risk-aversion has been implicitly discussed by Lin et al. in their work on reliable clustering [22], where the authors show that interpreting probabilities as weights does not result in good clusterings. Repetitive sampling from a large-scale uncertain graph in order to reduce the risk is inefficient. Motivated by this observation, Parchas et al. have proposed a heuristic to extract a good possible world in order to combine risk-aversion with efficiency [24]. However, their work comes with no guarantees. Jin et al. provide a risk-averse algorithm for distance

queries on uncertain graphs [18]. He and Kempe propose robust algorithms for the influence maximization problem [17]. Closest to our work lies the recent work by Tsourakakis et al. who studied the problem of finding efficiently risk averse graph and hypergraph matching algorithms [31].

Dense subgraph discovery (DSD) is a major graph mining topic, with numerous diverse applications, ranging from fraud detection to bioinformatics, see [15,30] for a detailed account of such applications. The *densest subgraph problem* (DSP) a popular DSD objective, that is solvable in polynomial time [13,16]. The DSP for undirected, weighted graphs $G(V, E, w), w : E \rightarrow \mathbb{R}^+$ maximizes the degree density $\rho(S) = \frac{w(S)}{|S|}$ over all possible subgraphs $S \subseteq V$, where $w(S) = \sum_{e \in e[S]} w(e)$ is the total induced weight by subgraph. In addition to the exact algorithm that is based on maximum flow computation, Charikar [8] proved that a greedy peeling algorithm produces a $\frac{1}{2}$-approximation of the densest subgraph in linear time, see also [20]. Galimberti et al. studied core decompositions – a concept intimately connected to DSD– on multilayer graphs [12]. Finally, Cadena et al. first studied DSD with negative weights [7], but their work focuses on anomaly detection, and the streaming nature of their input.

DSD on uncertain graphs is a less well studied topic. Zou was the first who discussed the DSP on uncertain graphs. His work shows –as expected– that the DSP in expectation can be solved in polynomial time [32]. The closest work related to our formulation is the recent work by Miyauchi and Takeda [23]. While their original motivation is also DSD on uncertain graphs, the modeling assumptions, and the mathematical objective differ significantly from ours. To the best of our knowledge, there is no work on risk-averse DSD under general probabilistic assumptions as ours.

3 Proposed Method

3.1 Why Negative Weights?

Risk-averse dense subgraph discovery. Uncertain graphs model the inherent uncertainty associated with graphs in a variety of applications. Here, we adopt the general model for uncertain graphs introduced by Tsourakakis et al. [31]. For completeness we present it in the following.

Model: Let $\mathcal{G}([n], E, \{g_e(\theta_e)\}_{e \in E})$ be an uncertain complete graph on n nodes, with the complete edge set $E = \binom{[n]}{2}$. The weight $w(e)$ (reward) of each edge $e \in E$ is drawn according to some probability distribution g_e with parameters $\boldsymbol{\theta_e}$, i.e., $w(e) \sim g_e(x; \boldsymbol{\theta_e})$. We assume that the weight of each edge is drawn independently from the rest; each probability distribution is assumed to have finite mean, and finite variance. Given this model, we define the probability of a given graph G with weights $w(e)$ on the edges as:

$$\mathbf{Pr}\left[G; \{w(e)\}_{e \in E}\right] = \prod_{e \in E} g_e(w(e); \boldsymbol{\theta_e}). \tag{1}$$

This model includes the standard Bernoulli model that is used extensively in the existing literature as a special case. Specifically, in the standard binomial uncertain graph model an uncertain graph is modeled by the triple $\mathcal{G} = (V, E, p)$ where $p : E \to (0, 1]$ is the function that assigns a probability of success to each edge independently from the other edges. According to the possible-world semantics [5,10] that interprets $\mathcal{G}$ as a set $\{G : (V, E_G)\}_{E_G \subseteq E}$ of $2^{|E|}$ possible deterministic graphs (worlds), each defined by a subset of E. The probability of observing any possible world $G(V, E_G) \in 2^E$ is

$$\mathbf{Pr}\left[G\right] = \prod_{e \in E_G} p(e) \prod_{e \in E \setminus E_G} (1 - p(e)).$$

A key property of these models to keep in mind, is that each edge e in the uncertain graph is independently distributed from the rest, and is associated with an expected reward μ_e (expectation) and a risk σ_e^2 (variance). Finally, observe that without any loss of generality in our general model described by equation (1) we have assumed that the edge set is $\binom{[n]}{2}$; non-edges can be modeled as edges with probability of existence zero.

Problem formulation. Intuitively, our goal is to find a subgraph $G[S]$ induced by $S \subseteq V$ such that its average reward $\frac{\sum_{e \in E(S)} w_e}{|S|}$ is large and the average associated risk is low $\frac{\sum_{e \in E(S)} \sigma_e^2}{|S|}$. To achieve this purpose we model the problem as a densest subgraph discovery problem in a graph with positive (reward) and negative (risk) edge weights. Specifically, for every edge $e = (u, v) \in E(G)$ we create two edges, a positive edge with weight equal to the expected reward, i.e., $w^+(e) = \mu_e$ and a negative edge with weight equal to the opposite of the risk of the edge, i.e., $w^-(e) = \sigma_e^2$. We wish to find a subgraph $S \subseteq V$ that has large positive average degree $\frac{w^+(S)}{|S|}$, and small negative average degree $\frac{w^-(S)}{|S|}$. We combine the two objectives into one objective $f : 2^V \to \mathbb{R}$ that we wish to maximize:

$$f(S) = \frac{w^+(S) + \lambda_1 |S|}{w^-(S) + \lambda_2 |S|}.$$

The parameters $\lambda_1, \lambda_2 \geq 0$ are positive reals. First, observe that this dense subgraph discovery formulation is applicable to any graph with positive and negative weights. Parameters λ_1, λ_2 allow us to control the size of the output as follows. Let us reparameterize the two parameters as $\lambda_1 = \rho\lambda, \lambda_2 = \lambda$. Then $f(S) = \frac{w^+(S) + \rho\lambda|S|}{w^-(S) + \lambda|S|}$, so if the ratio $\rho \geq 1$, then the objective favors larger node sets, whereas when $\rho < 1$ we favor smaller node sets.

We show how to solve the problem $\max_{S \subseteq V} f(S)$ by reducing it to standard dense subgraph discovery [16]. We perform binary search on $f(S)$ by answering queries of the following form:

> Does there exist a subset of nodes $S \subseteq V$ such that $f(S) \geq q$, where q is a query value?

Assuming an efficient algorithm for answering this query, and that the weights are polynomial functions of n, then using $O(\log n)$ queries we can find the optimal value for our objective $f : V \to \mathbb{R}$. By analyzing what each query corresponds to, we find:

$$\frac{w^+(S)+\lambda_1|S|}{w^-(S)+\lambda_2|S|} \geq q \rightarrow w^+(S)+\lambda_1|S| \geq q(w^-(S)+\lambda_2|S|) \rightarrow \quad (2)$$
$$\sum_{e\in E(S)} \underbrace{\Big(w^+(e)-qw^-(e)\Big)}_{\tilde{w}(e)} \geq |S| \underbrace{(q\lambda_2-\lambda_1)}_{q'} \rightarrow \sum_{e\in E(S)} \frac{\tilde{w}(e)}{|S|} \geq q'.$$

The latter inequality suggests that our original problem corresponds to querying in $\tilde{G}$, a modified version of G where the edge weight of any edge e becomes $w^+(e) - qw^-(e)$, whether there exists a subgraph S with density greater than q', where $q' = q\lambda_2 - \lambda_1$. However, this does not imply that our problem is poly-time solvable. The densest subgraph problem is poly-time solvable using a maximum flow formulation when the weights are positive rationals [16]. As we will prove in the next section, the densest subgraph problem when there exist negative weights is **NP**-hard in general. However, our analysis above leads to a straight-forward corollary that is worth stating. Intuitively, when for *each edge* e the ratio $\frac{w^+(e)}{w^-(e)}$ is large enough, then our problem is solvable in polynomial time.

Corollary 1. *Assume that* $w^+(e) \geq q_{max}w^-(e)$ *for all* $e \in E^+ \cup E^-$, *where* q_{max} *is the maximum possible query value. Then, the densest subgraph problem is solvable in polynomial time.*

Proof. If $w^+(e) \geq q_{max}w^-(e)$ for each $e \in E$, we obtain $\tilde{w}(e) \geq 0$ for each $e \in E$ in inequality (2) is equivalent to solving the densest subgraph problem in an undirected graph with non-negative weights, see [16,29].

Observe that a trivial upper bound of $q_{\max}$ can be obtained by setting $w^+(S) = \sum_{e\in E(G)} w^+(e), w^-(S) = 0$, and since $\lambda_1|S| \leq \lambda_1 n, \lambda_2|S| \geq \lambda_2$ for all $S \neq \emptyset$, we see that $q_{\max} \leq \frac{\sum_{e\in E(G)} w^+(e)+\lambda_1 n}{\lambda_2}$. For polynomially bounded weights, this is a polynomial function of n, hence the number of binary search iterations is logarithmic.

Controlling the risk in practice. There exist real-world scenarios where the practitioner wants to control the trade-off between reward and risk, see [31]. An effective way to change the risk tolerance is as follows by multiplying the negative induced weight $w^-(S)$ by $B \in (0, +\infty)$. Namely, our objective $f : 2^V \to \mathbb{R}$ is $f(S) = \frac{w^+(S)+\lambda_1|S|}{Bw^-(S)+\lambda_2|S|}$. An interesting open problem is to develop a formal (bi-criteria) approximation for risk averse DSD along the lines of [26,31].

Soft and hard exclusion dense subgraph queries. Given the Twitter network, where user accounts may interact in more than one ways (e.g., *follow, retweet, mention, quote, reply*), can we find a dense subgraph that does not

contain any *follow* but contains many *reply* interactions? We ask this question in a more general form.

Problem 3. Given a large-scale multilayer network, how do we find a dense subgraph that *excludes* certain types of edges?

We consider two types of such queries, the *soft* and *hard* queries. In the former case we want to find subgraphs with perhaps few edges of certain types, in the latter case we want to exclude fully such edges. An algorithmic primitive that can answer efficiently these queries can be used to understand the structure of large-scale multilayer networks, and find anomalies and interesting patterns. In principle, we set the edge weight of an excluded type to $-W$ where $W > 0$ is an input parameter. If we set $W = +\infty$, subgraphs that do not induce any edge of any excluded type will have positive weight, whereas subgraphs that induce even one edge of a forbidden type will have $-\infty$ weight. If we set W to a small value, then there may be some undesired edges in the output subgraph. The pseudo-code in Algorithm 1 describes this approach.

Algorithm 1: `Exclusion-Queries`

Input: $G(V, E), \{\text{labels}\}, W > 0$
for $e \in E(G)$ **do**
  for $c \in$ *labels* **do**
    if *If* $type(e) = c$ **then**
      $w(e) \leftarrow -W$ (else $w(e)$ remains 1)
    end
  end
end
return $S \subseteq V$ *that achieves maximum average degree in* $G(V, E, w)$.

3.2 Hardness

We prove that solving the densest subgraph problem on graphs with negative weights is **NP**-hard. We formally define our problem Neg-DSD.

Problem 4 (Neg-DSD). Given a graph G with loops and possibly negative weights, find the subset A of V that maximizes $\frac{w(A)}{|A|}$.

We prove that Neg-DSD is **NP**-hard. Our reduction is based on the the proposed strategy by Peter Shor for showing that the max-cut problem on graphs with possibly negative edges is **NP**-hard [1]. This is stated as the Theorem 1. For convenience, we also define the decision version of the maximum cut problem [1].

Theorem 1. Neg-DSD is **NP**-hard.

Problem 5 (Max-Cut). Given a graph $G(V, E)$ and a constant c, find a partition (V_1, V_2) of V such that $cut(V_1, V_2) = \sum_{u \in V_1, v \in V_2} w(u, v) > c$.

Our proof strategy is inspired by Peter Shor's proof that max-cut with negative weight edges is **NP**-hard [1]. We provide a detailed proof sketch.

Proof. First, we define the Positive-Cut problem, and show that it is **NP**-hard by reducing the Max-Cut problem to it.

Problem 6 (Positive-Cut). Given a graph G with possibly negative weights, find a partition (V_1, V_2) of V such that $cut(V_1, V_2) > 0$.

We choose two nodes u, v that lie on opposite sides of an optimal max cut (V_1^*, V_2^*). Despite the fact we do not know the max cut, we can perform this step in polynomial time by repeating the following procedure for all possible pairs of nodes; if we cannot find a positive cut for any of the pairs, then the answer to the Max-Cut is negative. With a vary large value d (e.g. $d = 1 + \max_{e \in E}(w^+(e))$), we construct a graph G' by adding negative weight equal to $-d$ from u and v to all other vertices, and an edge of weight $(n-2)d - c$ between u, v. All cuts that place u, v on the same side will be negative in G' provided d is sufficiently large. All other cuts will be positive if and only if the corresponding cut in G is greater than c. Therefore, Positive-Cut is **NP**-hard.

Finally we prove that Neg-DSD is **NP**-hard using a reduction from Positive-Cut. We construct a graph G' by negating every weight in G putting a loop on every vertex so that its weighted degree is zero. Therefore, the sum of the degrees of any set A in G' is equal to $\sum_{v \in A} 0 = 2w(A) + cut(A, V \backslash A)$. Finally, observe that a cut (V_1, V_2) has positive weight in G if and only if V_1 has positive average degree.

3.3 Algorithms and Heuristics

A popular algorithm for the densest subgraph problem is Charikar's algorithm [8]. We study the performance of this algorithm in the presence of negative weights. The pseudocode is given as Algorithm 2. The algorithm iteratively removes from the graph the node of the smallest degree $d(v) = deg^+(v) - deg^-(v)$, and among the sequence of n produced graphs, outputs the one that achieves the highest degree density. Our main theoretical result for the performance of Algorithm 2 is stated as Theorem 2.

Theorem 2. Let $G(V, E, w)$, $w : E \to \mathbb{R}$ be an undirected weighted graph with possibly negative weights. Let S^* be the optimal densest subgraph in G with average density $\frac{w(S^*)}{|S^*|} = \rho^*$. If the negative degree $deg^-(u)$ of any node u is upper bounded by Δ, then Algorithm 2 outputs a set whose density is at least $\frac{\rho^*}{2} - \frac{\Delta}{2}$.

Proof. By the optimality of S^* we obtain that $d_{S^*}(v) \geq \rho^*$, and then trivially $deg^+(v) \geq \rho^*$. Consider the execution of algorithm 2, and let $u \in S^*$ be the first vertex from S^* removed during the peeling. Let S be the set of nodes at that

Algorithm 2: `Peeling`

Input: G
$n \leftarrow |V|, H_n \leftarrow G$;
for $i \leftarrow n$ *to* 2 **do**
  Let v be the vertex of G_i of minimum degree, i.e.,
  $d(v) = deg^+(v) - deg^-(v)$ (break ties arbitrarily);
  $H_{i-1} \leftarrow H_i \backslash v$;
end
return H_j *that achieves maximum average degree among* H_i*s,* $i = 1, \ldots, n$.

iteration, including u. By the peeling process, we have $d_S(v) \geq d_S(u)$ for all $v \in S$. Furthermore,

$$d_S(u) = deg_S^+(u) - deg_S^-(u) \geq \deg_S^+(u) - \Delta,$$

since by our assumption $deg_S^-(u) \leq deg^-(u) \leq \Delta$. This implies that

$$2w(S) = \sum_{v \in S} d_S(v) \geq \sum_{v \in S} deg_S^+(v) - |S|\Delta \geq |S|(\rho^* - \Delta) \rightarrow \frac{w(S)}{|S|} \geq \frac{\rho^*}{2} - \frac{\Delta}{2}.$$

This yields that the output of Algorithm 2 outputs a set of nodes $\bar{S}$ with density at least $\frac{\rho^*}{2} - \frac{\Delta}{2}$.

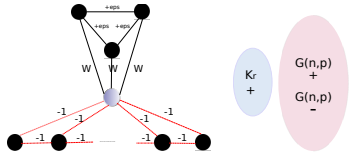
When the additive error term in the approximation is small compared to the term $\frac{\rho^*}{2}$, then the peeling algorithm performs effectively with strong guarantee. In practice, Algorithm 2 performs well on large-scale graphs where the negative weights are small. In the presence of very large negative weights, the approximation guarantees become meaningless.

Claim. In the presence of large negative weights, Algorithm 2 may perform arbitrarily bad.

This is illustrated in Figure 1(a) that provides an instance of a graph with large negative weights. Intuitively, in bad instances when there exist large negative degrees, nodes that should not be removed early on by the peeling process, are actually removed. Specifically, consider when $W = \frac{n-4}{3}$, then $3W - n < -3$. The degrees of nodes are

$$\underbrace{3W - n}_{\text{one node}} < \underbrace{-3}_{n-2 \text{ nodes}} < \underbrace{-2}_{\text{two nodes}} < 0 < \underbrace{2\epsilon + W}_{\text{three nodes}}.$$

Therefore, the center node is removed first, and the peeling algorithm will output as the densest subgraph the triangle of density ϵ. The optimal densest subgraph has $\frac{3W+3\epsilon}{4}$. By allowing ϵ to be arbitrarily small, we observe that the approximation ratio becomes arbitrarily bad. To tackle such scenarios, i.e., where nodes from the densest subgraph are peeled earlier than when they should, we



(a) (b)

Fig. 1. Bad peeling instances. For details, see Section 3.

propose an effective heuristic which is outlined in Algorithm 3. The algorithm again peels the nodes but scores every node u according to $Cdeg^+(u) - deg^-(u)$, where $C > 0$ is a parameter that is part of the input.

Remark about C in Algorithm 3. While Figure 1(a) suggests the use of $C \geq 1$, it could be the case that C has to be set to a value less than 1 to obtain good results. We provide an example where using $C < 1$ can help in providing a better peeling permutation of the nodes. Consider a graph whose weights are either $+1$ or -1, that consists of two connected components. The first component is a positive clique on r nodes. The second component is the union of two random binomial graphs $G(n, p)$ where $p = \frac{1}{2}$. This is illustrated in Figure 1(b). The degree of any node u in the first component is $deg(u) = deg^+(u) - deg^-(u) = (r-1) - 0$. The expected degree of any node in the second component is 0. Furthermore, the average degree of any subset of nodes in the 2nd component is 0 in expectation. However, using concentration bounds (details omitted) one can show that it is likely that there will exist a node u in the second component with positive degree $\kappa\sqrt{n}$ and negative degree $\kappa'\sqrt{n}$ with $\kappa > \kappa'$, and therefore positive total degree. Only the use of a $C < 1$ will improve the peeling ordering; e.g., in the extreme case where $C = 0$ the nodes of the second component will be removed first.

Rule-of-thumb. In practice, given that each run of the algorithm takes linear time, we can afford to run the algorithm for a bunch of C values and return the densest subgraph among the outputs produced by each run, instead of using one value for C. This strategy is applied in Section 4.

Algorithm 3: `Heuristic-Peeling`

Input: $G, C \in (0, +\infty)$
$n \leftarrow |V|, H_n \leftarrow G$ **for** $i \leftarrow n$ *to* 2 **do**
  Let v be the vertex of G_i of minimum degree, i.e.,
  $d(v) = Cdeg^+(v) - deg^-(v)$ (break ties arbitrarily) $H_{i-1} \leftarrow H_i \backslash v$
end
return H_j *that achieves maximum average degree among* H_i*s,* $i = 1, \ldots, n$.

Shifting the negative weights. Finally, for the sake of completeness, we mention that the perhaps natural idea of shifting all the weights by the most negative weight in the graph, in order to obtain non-negative weights, and apply

the exact polynomial time algorithm on the weight-shifted graph may perform arbitrarily bad. To see why, consider a graph that consists of three components, a triangle with positive weights equal to 1, an edge with a large negative weight $-\Delta < 0$, and a large clique with small negative weight $-\epsilon < 0$. In this graph, the densest subgraph is the positive triangle, but by shifting the weights by $+\Delta$, there exists values for ϵ, Δ such that the densest subgraph in the weight-shifted graph is the negative clique. Also experimentally, this heuristic performs extremely poorly.

4 Experimental results

4.1 Experimental setup

Datasets. All the datasets we have used in our experiments are publicly available, and are described in Table 2. We use four protein-protein interaction uncertain graphs, *Biogrid, Collins, Gavin, Krogan* that have been used in prior biological studies (e.g., [9,14,21]) and are available at [2]. The set of nodes represents proteins and the probability of the edge is equal to the existence probability of the interaction. Another uncertain graph, available at [4], is created from the TMDB movie database as follows. The set of nodes corresponds to actors, and the probability of the edge is equal to the probability that these two actors co-star in a movie. Specifically, for actors u, v, the probability $p(u, v)$ is equal to the Jaccard coefficient $J(M_u, M_v) = \frac{|M_u \cap M_v|}{|M_u \cup M_v|}$, where M_u, M_v are the sets of movies that u, v have co-starred. We choose weights to represent a function of the popularity of the movies, i.e., a score assigned to each movie by TMDB (1 is the lowest score in our collected dataset). Intuitively, these scores reflect the reward of a potential collaboration between two actors. While there are many ways to set the weight of an edge for two actors (e.g. average popularity), we focus on the most popular movies they have co-starred in. The main rationale behind this choice is that the majority of actors play in movies whose majority popularity is 1, i.e., the lowest possible. For a pair of actors $\{u, v\}$, let $s_0 \geq \ldots \geq s_{k-1}$ where $k = \min(|M_u \cap M_v|, 5)$ be the popularity scores of movies they have co-starred in. We set $w(u, v) = \sum_{j=0}^{k-1} \frac{s_j}{2^j}$, i.e., a discounted sum of popularities, focusing more on the most popular movies the two actors have co-starred in.

Finally, we used an open-source twitter API crawler to monitor twitter traffic between February 1st and February 14th, 2018 [25]. We provided detailed information about each daily graph. Here, the number of edges is a five dimensional vector, whose coordinates correspond to the number of follows, mentions, retweets, quotes, and replies respectively. We will make the Twitter datasets available upon proper anonymization. The datasets we use are overall small, and medium sized, therefore our proposed algorithm for a fixed C value, requires few seconds or few minutes for the largest graphs.

Unfortunately, due to space constraints we present a representative sample of our findings. The interested reader can find an extended version of this paper online [3].

Table 2. Datasets used in our experiments. The number of vertices n and edges m is recorded for each graph. The datasets annotated by ⊙ have been created by us, and are publicly available. For details, see Section 4.1.

Name	n	m
■ Biogrid	5 640	59 748
■ Collins	1 622	9 074
■ Gavin	1 855	7 669
■ Krogan core	2 708	7 123
■ Krogan extended	3 672	14 317
⊙ TMDB	160 784	883 842
⊙ Twitter (Feb. 1)	621 617	(902 834, 387 597, 222 253, 30 018, 63 062)
⊙ Twitter (Feb. 2)	706 104	(1 002 265, 388 669, 218 901, 29 621, 64 282)
⊙ Twitter (Feb. 3)	651 109	(1 010 002, 373 889, 218 717, 27 805, 59 503)
⊙ Twitter (Feb. 4)	528 594	(865 019, 435 536, 269 750, 32 584, 71 802)
⊙ Twitter (Feb. 5)	631 697	(999 961, 396 223, 233 464, 30 937, 66 968)
⊙ Twitter (Feb. 6)	732 852	(941 353, 407 834, 239 486, 31 853, 67 374)
⊙ Twitter (Feb. 7)	742 566	(1 129 011, 406 852, 236 121, 30 815, 68 093)

Machine specs and code. The experiments were performed on a single machine, with an Intel Xeon CPU at 2.83 GHz, 6144KB cache size, and 50GB of main memory. The code is written in Python, and available at `https://github.com/negativedsd`.

4.2 Risk-averse DSD

We perform two risk averse DSD experiments. First, for various fixed pairs of (λ_1, λ_2) values, we range the parameter B (reminder: B is the multiplicative factor of $w^-(S)$, see Controlling the risk in practice, Section 3.1) to control the trade-off between expected average reward and average risk. A typical outcome of our algorithm on the set of uncertain graphs we have tested it on for $\lambda_1 = \lambda_2 = 1$, and $C = 1$ is summarized in Table 3. As B increases, we tolerate less risk, but the expected average reward drops as well.

Table 3. Exploring the effect of risk tolerance parameter B on the *gavin* dataset. For details, see Section 4.2.

B	Average exp. reward	average risk	$\lvert S^* \rvert$
0.25	0.18	0.09	6
1	0.17	0.08	10
2	0.13	0.06	31

In our second experiment we test the effect of rest of the parameters. We fix $B = 1$, and then we perform the following procedure. For each dataset, we fix a pair of (λ_1, λ_2) values and run our proposed algorithm using 7 values of C. The

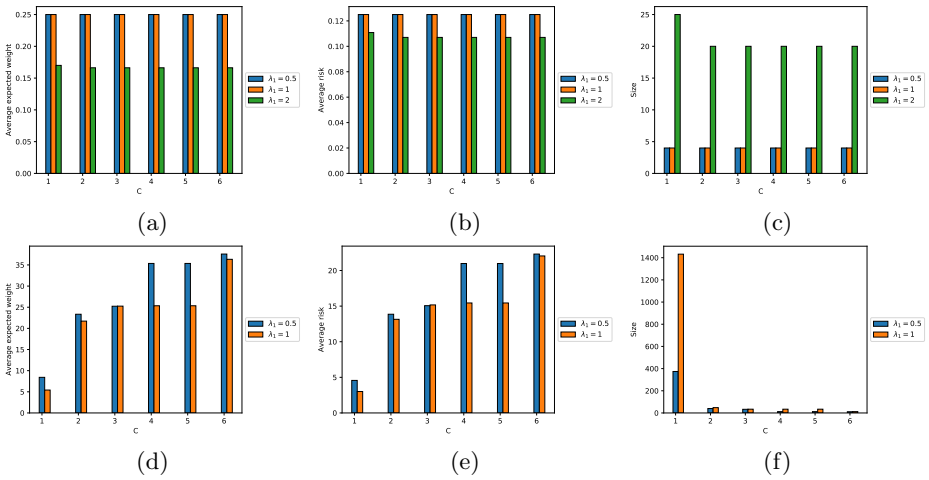


Fig. 2. Risk averse DSD results for Biogrid (a) average expected weight, (b) average risk, (c) output size, and for TMDB (d) average expected weight, (e) average risk, (f) output size. For details, see Section 4.2.

C value 0.5 always resulted in trivial results that would skew a lot the plots so it is omitted. Specifically, for $C = 0.5$ for all three pairs of λ values we use, we obtain (almost) the whole graph as output of the peeling process. The three pairs of λ values we use are $(\lambda_1, \lambda_2) \in \{(0.5, 1), (1, 1), (2, 1)\}$. Our results are shown in Figure 2. For the TMDB graph, the last pair of λ values results in obtaining the whole graph as the optimal solution, so we omit it from Figures 2(d), (e), and (f). We include these two datasets as they show that the change in the C value in principle does not affect risk aversion (Figure 2(b)), but it could happen due to the different peeling orderings it produces that the output will be associated with different risk (Figure 2)(d)). We also observe that as we increase λ_1 the size of the output increases. This agrees with the insights we provide in Section 3; namely, we reward larger sets of nodes.

4.3 Mining Twitter using DSD-Exclusion queries

We test our DSD exclusion query primitive on the Twitter daily data. We present results that we obtain for different pairs of graphs induced by different types of interactions, for $C = 1$. For each such pair, we perform all possible non-trivial exclusion queries:

- Every type of interaction is allowed (query denoted as $[1, 1]$).
- One of the two interaction types is excluded (queries denoted as $[1, 0]$, and $[0, 1]$ for excluding the first and second type of interactions respectively).

Figure 3 shows for each pair of interactions the degree density (1st row), and the output size (2nd row). Interestingly, observe that in Figure 3(c) the exclusion query $[0, 1]$ that excludes mentions and allows retweets results in density close

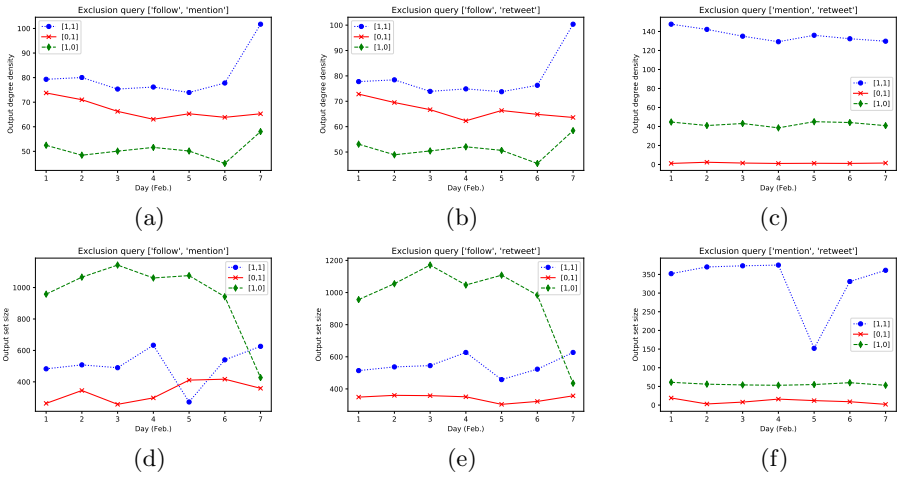


Fig. 3. Degree density (1st row), and output size(2nd row) for three exclusion queries per each pair of interaction types over the period of the first week of February 2018. (a),(d) Follow and mention. (b),(e) Follow and retweet. (c),(f) Mention and retweet.

to 0. This is because the Twitter API considers every retweet as a mention. By excluding mentions, we exclude all retweets. The density is not exactly zero, due to some small noise in the crawled mentions, i.e., there exist a few retweets that have not been included in the mentions. We have performed more exclusion queries that involve more types of interactions. For instance, by looking into *reply, quote, retweet* interactions, we find the following results for two queries on February 1st, 2018.

- When we allow all types we find a subset of 351 nodes, whose retweet density is 72.6, reply density 3.86, and quote density 1.08. We observe this difference since the retweet layer of interactions is much denser than the other two.
- When we exclude the retweets, but allow quotes and replies, we find a set of 30 nodes whose reply degree density is 15.46, and quote degree density 0.066.

Effect of C, and W. As we discussed earlier, ranging W, from small values to $+\infty$ quantifies how much we care about excluding the undesired edge types. Table 4 shows what we observe typically on all experiments we have performed. Specifically, we perform an exclusion query $[1, 0]$ on the *retweet, reply* interactions. We denote by S^* the output of Algorithm 3. By inspecting the last column $\rho_{\text{reply}}(S^*)$ of the table, we observe that even when we set the weight of each reply interaction to -1 (soft query), our algorithm outputs a set S^* with very few replies, for all $C \in \{\frac{1}{10}, 1, 10\}$ values we use. When W is set to the very large value 200 000 (hard query), $\rho_{\text{reply}}(S^*)$ becomes 0 but we also observe a drop in the degree density of the retweets. For instance for $C = 1$, $\rho_{\text{retweet}}(S^*)$ drops from 72.70 to 30.38.

Table 4. Exploring the effect of the negative weight $-W$ on the excluded edge types for various C values. For details, see Section 4.3.

C	W	$\lvert S^*\rvert$	$\rho_{\text{retweet}}(S^*)$	$\rho_{\text{reply}}(S^*)$
0.1	1	296	63.44	-0.75
	5	99	45.67	-0.01
	200 000	200	30.37	0
1	1	346	72.70	-2.75
	5	319	68.70	-1.29
	200 000	200	30.38	0
10	1	351	73.10	-3.31
	5	351	73.10	-3.31
	200 000	200	30.37	0

5 Open Problems

In this work we have initiated a formal study of DSD with negative weights. Understanding better the complexity of the problem remains open. For example, we provided sufficient conditions under which the problem is poly-time solvable. Developing an approximation or bi-criteria approximation algorithms for risk averse DSD that aims to maximize the expected reward subject to bounds on the risk is also an interesting open problem. Finally, and more broadly, designing risk-averse, efficient graph mining algorithms is an interesting direction.

References

1. Max-cut with negative weight edges by Peter Shor `https://cstheory.stackexchange.com/questions/2312/max-cut-with-negative-weight-edges`.
2. `http://www.paccanarolab.org/static_content/clusterone/cl1_datasets.zip`.
3. Submitted to ECML-PKDD 2019, full version available at `https://lastinggems.files.wordpress.com/2019/04/neg-dsd.pdf`.
4. Tmdb uncertain graph. `https://drive.google.com/open?id=1C69MndtfSoUflPkeBa0mbiC9FZD6xbpN`.
5. B. Bollobás, S. Janson, and O. Riordan. The phase transition in inhomogeneous random graphs. *Random Structures & Algorithms*, 31(1):3–122, 2007.
6. F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich. Core decomposition of uncertain graphs. In *Proc. of the 20th ACM SIGKDD conference*, pages 1316–1325. ACM, 2014.
7. J. Cadena, A. K. Vullikanti, and C. C. Aggarwal. On dense subgraphs in signed network streams. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 51–60. IEEE, 2016.
8. M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, 2000.
9. S. R. Collins, P. Kemmeren, X.-C. Zhao, J. F. Greenblatt, F. Spencer, F. C. Holstege, J. S. Weissman, and N. J. Krogan. Toward a comprehensive atlas of the physical interactome of saccharomyces cerevisiae. *Molecular & Cellular Proteomics*, 6(3):439–450, 2007.

10. N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.
11. E. Galimberti, F. Bonchi, and F. Gullo. Core decomposition and densest subgraph in multilayer networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1807–1816. ACM, 2017.
12. E. Galimberti, F. Bonchi, F. Gullo, and T. Lanciano. Core decomposition in multilayer networks: Theory, algorithms, and applications. *arXiv preprint arXiv:1812.08712*, 2018.
13. G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *Journal of Computing*, 18(1), 1989.
14. A.-C. Gavin, P. Aloy, P. Grandi, R. Krause, M. Boesche, M. Marzioch, C. Rau, L. J. Jensen, S. Bastuck, B. Dümpelfeld, et al. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 440(7084):631, 2006.
15. A. Gionis and C. E. Tsourakakis. Dense subgraph discovery: Kdd 2015 tutorial. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2313–2314. ACM, 2015.
16. A. V. Goldberg. Finding a maximum density subgraph. Technical report, University of California at Berkeley, 1984.
17. X. He and D. Kempe. Robust influence maximization. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 885–894, New York, NY, USA, 2016. ACM.
18. R. Jin, L. Liu, and C. C. Aggarwal. Discovering highly reliable subgraphs in uncertain graphs. In *Proceedings of KDD 2011*, pages 992–1000, 2011.
19. D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of KDD 2003*, pages 137–146. ACM, 2003.
20. S. Khuller and B. Saha. On finding dense subgraphs. In *36th International Colloquium on Automata, Languages and Programming (ICALP)*, 2009.
21. N. J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A. P. Tikuisis, et al. Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, 440(7084):637, 2006.
22. L. Liu, R. Jin, C. Aggarwal, and Y. Shen. Reliable clustering on uncertain graphs. In *Proceedings of ICDM 2012*, pages 459–468. IEEE, 2012.
23. A. Miyauchi and A. Takeda. Robust densest subgraph discovery. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1188–1193. IEEE, 2018.
24. P. Parchas, F. Gullo, D. Papadias, and F. Bonchi. The pursuit of a good possible world: extracting representative instances of uncertain graphs. In *Proceedings SIGMOD 2014*, pages 967–978, 2014.
25. P. Pratikakis. twawler: A lightweight twitter crawler. *arXiv preprint arXiv:1804.07748*, 2018.
26. R. Ravi and M. X. Goemans. The constrained minimum spanning tree problem. In *Scandinavian Workshop on Algorithm Theory*, pages 66–75. Springer, 1996.
27. A. E. Roth, T. Sönmez, and M. U. Ünver. Kidney exchange. *The Quarterly Journal of Economics*, 119(2):457–488, 2004.
28. J. Serra and J. L. Arcos. An empirical evaluation of similarity measures for time series classification. *Knowledge-Based Systems*, 67:305–314, 2014.
29. C. E. Tsourakakis. The k-clique densest subgraph problem. *24th International World Wide Web Conference (WWW)*, 2015.
30. C. E. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013

31. C. E. Tsourakakis, S. Sekar, J. Lam, and L. Yang. Risk-averse matchings over uncertain graph databases. *arXiv preprint arXiv:1801.03190*, 2018.
32. Z. Zou. Polynomial-time algorithm for finding densest subgraphs in uncertain graphs. In *Proceedings of MLG Workshop*, 2013.