Community-aware network sparsification

Aristides Gionis Polina Rozenshtein Nikolaj Tatti
Aalto University
Espoo, Finland
firstname.lastname@aalto.fi

Evimaria Terzi
Boston University
Boston, USA
evimaria@bu.edu

Abstract

Network sparsification aims to reduce the number of edges of a network while maintaining its structural properties: shortest paths, cuts, spectral measures, or network modularity. Sparsification has multiple applications, such as, speeding up graph-mining algorithms, graph visualization, as well as identifying the important network edges.

In this paper, we consider a novel formulation of the network-sparsification problem. In addition to the network, we also consider as input a set of communities. The goal is to sparsify the network so as to preserve the network structure with respect to the given communities. We introduce two variants of the community-aware sparsification problem, leading to sparsifiers that satisfy different *connectedness* community properties. From the technical point of view, we prove hardness results and devise effective approximation algorithms. Our experimental results on a large collection of datasets demonstrate the effectiveness of our algorithms.

1 Introduction

Large graphs, or networks, arise in many applications, e.g., social networks, information networks, and biological networks. Real-world networks are usually sparse, meaning that the actual number of edges in the network m is much smaller than $\mathcal{O}(n^2)$, where n is the number of network nodes. Nonetheless, in practice, it is common to work with networks whose average degree is in the order of hundreds or thousands, leading to many computational and data-analysis challenges.

Sparsification is a fundamental operation that aims to reduce the number of edges of a network while maintaining its structural properties. Sparsification has numerous applications, such as, graph summarization and visualization, speeding up graph algorithms, and identification of important edges. A number of different sparsification methods have been proposed, depending on the network property that one aims to preserve. Typical properties include paths and connectivity [7,21], cuts [1,9], and spectral properties [2,6,19].

Existing work on network sparsification ignores the fact that the observed network is the result of different latent factors. For instance, imagine a Facebook user who posts a high-school photo, which leads to a discussion thread among old high-school friends. In this case, the participation of users in a discussion group is a result of an underlying community. In general, the network structure reflects a number of underlying (and potentially overlapping) communities. Thus, if it is this community structure that guides the network-formation process, then the community structure should also be taken into consideration in network sparsification.

Motivated by this view, we consider the following problem: *Given a network and a set of potentially overlapping communities, sparsify the network so as to preserve its structural properties with respect to the given communities.* Our goal is to find a small set of edges that best summarize, or explain, a given community structure in the network.

Our setting has many interesting applications. For example, consider a social network where users discuss various topics. Each topic defines a community of people interested in the topic. Given a set of topics, we want to find a sparse network that best explains the respective communities. Similar problems arise in collaboration networks, where communities are defined by collaboration themes, consumer networks where communities are defined by products, etc. Finding an optimal sparse network with respect to a set of communities is a means of understanding the interplay between network structure and content-induced communities.

We formalize the above intuition by defining the NetSparse problem: given an undirected graph $G = (V,E)$ and a set of communities $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ over V, we ask to find a sparsified graph $G' = (V',E')$ with $V' = \cup_{i=1,\ldots,\ell} C_i$ and $E' \subseteq E$, so as to minimize $|E'|$ and guarantee that every graph $G'[C_i]$, induced by the nodes in the community C_i, satisfies a certain *connectedness requirement*.

Different connectedness requirements give rise to different variants of the NetSparse problem. We consider three such requirements: (*i*) *connectivity*, (*ii*) *density* and (*iii*) *star containment*. While connectivity has been addressed by previous work [4], we are the first to introduce and study the other two properties, which define the SparseDens and SparseStars problems, respectively. In the SparseDens problem the requirement is that each induced graph $G'[C_i]$ has a minimum density requirement. In the SPARSESTARS problem the requirement is that $G'[C_i]$ contains a star as a subgraph. We establish the computational complexity of the two problems, SPARSEDENS and SPARSESTARS, and present approximation algorithms for solving them.

An interesting special case arises when the input to our problem consists only of the collection of communities and there is no network $G = (V,E)$. In this case, we can consider that G is the complete graph (clique) and the NETSPARSE becomes a *network design* problem, where the goal is to construct a network that satisfies the connectedness requirement among the nodes in the different communities.

The list of our contributions is the following.

- We introduce the novel problem of sparsifying a network while preserving the structure of a given set of communities.
- We formulate different variants of this *network-aware sparsification* task, by considering preserving connectedness properties within communities.
- For the proposed formulations we present complexity results and efficient approximation algorithms.
- We present experimental results on a large collection of real datasets, demonstrating that our algorithms effectively sparsify the underlying network while maintaining the required community structure and other key properties of the original graph.

We note that our implementation and datasets will be publicly available. Proofs, other results, and additional experiments are in the supplementary material.

2 General problem definition

Our input consists of an underlying undirected graph $G = (V,E)$ having $|V| = n$ vertices and $|E| = m$ edges. As a special case, the underlying network G can be *empty*, i.e., there is no underlying network at all. We treat this case equivalently to the case in which the underlying network is the complete graph (clique).

Additionally, we consider as input a collection of ℓ sets $\mathcal{C} = \{C_1,\ldots,C_\ell\}$ over V, i.e., $C_i \subseteq V$. We think of the sets $\mathcal{C}$ and we refer to them as network *communities*. We assume that the sets in $\mathcal{C}$ may be overlapping.

Our objective is to find a *sparsifier* of the network G that maintains certain *connectedness* properties with respect to the given communities $\mathcal{C}$. A sparsifier of G is a subgraph $G' = (V',E')$, where the number of edges $|E'|$ is significantly smaller than $|E|$. The vertices V' spanned by G' are the vertices that appear in at least one community C_i, i.e., $V' = \cup_{i=1}^{\ell} C_i$. Without loss of generality we assume that $\cup_{i=1}^{\ell} C_i = V$, so $V' = V$.

Connectedness properties: To formally define the sparsification problem, we need to specify what it means for the sparse network to satisfy a connectedness property with respect to the set of communities $\mathcal{C}$.

We provide the following formalization: given a graph $G = (V,E)$ and $S \subseteq V$, we use $E(S)$ to denote the edges of E that have both endpoints in S, and $G(S) = (S, E(S))$ is the subgraph of G *induced* by S. We are interested in whether a graph $G = (V,E)$ satisfies a certain property ρ for a given set of communities $\mathcal{C} = \{C_1,\ldots,C_\ell\}$ where $C_i \subseteq V$. We say that G satisfies property ρ with respect to a community C_i if the induced subgraph $G(C_i)$ satisfies property ρ. We write $\mathbb{I}_\rho(G,C_i) = 1$ to denote the fact that $G(C_i)$ satisfies property ρ, and $\mathbb{I}_\rho(G,C_i) = 0$ otherwise.

We consider three graph properties: (*i*) *connectivity*, denoted by c; (*ii*) *density*, denoted by d; and (*iii*) *star containment*, denoted by s. The corresponding indicator functions are denoted by $\mathbb{I}_c$, $\mathbb{I}_{d\geq\alpha_i}$, and $\mathbb{I}_s$.

The connectivity property requires that each set C_i induces a connected subgraph. The density property requires that each set C_i induces a subgraph of density at least α_i. The density property is motivated by the common perception that communities are usually densely connected. The star-containment property requires that each set C_i induces a graph that contains a star. The intuition is that star-shaped communities have small diameter and also have a community "leader," which corresponds to the center of the graph.

Problem definition: We can now define the general problem that we study in this paper.

PROBLEM 1. (NETSPARSE) *Consider a network $G = (V,E)$, and let ρ be a graph property. Given a set of ℓ communities $\mathcal{C} = \{C_1,\ldots,C_\ell\}$, we want to find a* sparse network *$G' = (V,E')$ so that (i) $E' \subseteq E$; (ii) G' satisfies property ρ for all communities $C_i \in \mathcal{C}$; and (iii) the total number of edges (or total edge weight, if defined) on the sparse network $|E'|$ is minimized.*

One question is whether a feasible solution for problem NETSPARSE exists. This can be easily checked by testing if property ρ is satisfied for each C_i in the original network G. If this is true, then a feasible solution exists — the original network G is such a solution. Furthermore, if property ρ is not satisfied for a community C_i in the original network, then this community can be dropped, and a feasible solution exists for all the communities for which the property is satisfied in the original network.

One should also note that the problem complexity and the algorithms for solving Problem 1 depend on the property ρ. This is illustrated in the next paragraph, as well as in the next two sections.

Connectivity. Angluin et al. [4] study the NetSparse problem for the connectivity property. They show that it is an **NP**-hard problem and provide an algorithm with logarithmic approximation guarantee.

3 Sparsification with density constraints

Density
↓
$\frac{|e(c_i,c_i)|}{|C_i|C_2}$

We assume that each community $C_i \in \mathcal{C}$ is associated with a *density requirement* α_i, where $0 \leq \alpha_i \leq 1$. This is the target density for community C_i in the sparse network. As a special case all communities may have the same target density, i.e., $\alpha_i = \alpha$. We say that a network $G' = (V,E')$ satisfies the density property with respect to a community C_i and density threshold α_i, if $|E'(C_i)| \geq \alpha_i \binom{|C_i|}{2}$, that is, the density of the subgraph induced by C_i in G' is at least α_i. We denote this by $\mathbb{I}_{d\geq\alpha_i}(G',C_i) = 1$; otherwise we set $\mathbb{I}_{d\geq\alpha_i}(G',C_i) = 0$.

The SparseDens problem is defined as the special case of Problem 1, where ρ is the density property. Before presenting our algorithm for the SparseDens problem, we first establish its complexity.

Proof in supplementary material

Proposition 3.1. *The* SparseDens *problem is* **NP***-hard.*

We now present DGreedy, a greedy algorithm for the SparseDens problem. Given an instance of SparseDens, i.e., a network $G = (V,E)$, a set of ℓ communities C_i, and corresponding densities α_i, the algorithm provides an $\mathcal{O}(\log \ell)$-approximation guarantee.

To analyze DGreedy, we consider a *potential function* Φ, defined over subsets of edges of E. For an edge set $H \subseteq E$, a community C_i, and density constraint α_i, the potential Φ is defined as

$$\Phi(H, C_i) = \min\left\{0, |H(C_i)| - \left\lceil \alpha_i \binom{|C_i|}{2} \right\rceil \right\}, \tag{3.1}$$

where, $\lceil\cdot\rceil$ denotes the ceiling function. Note that

$$\begin{aligned}\Phi(H,C_i) < 0 \quad &\text{if} \quad \mathbb{I}_{d\geq\alpha_i}(G(C_i,H),C_i) = 0, \quad \text{and}\\ \Phi(H,C_i) = 0 \quad &\text{if} \quad \mathbb{I}_{d\geq\alpha_i}(G(C_i,H),C_i) = 1.\end{aligned}$$

In other words, Φ is negative if the edges H do not satisfy the density constraint on C_i, and becomes zero as soon as the density constraint is satisfied.

We also define the *total potential* of a set of edges H with respect to the input communities $\mathcal{C}$ as

$$\Phi(H) = \sum_{C_i \in \mathcal{C}} \Phi(H,C_i). \tag{3.2}$$

The choices of DGreedy are guided by the potential function Φ. The algorithm starts with $E' = \emptyset$ and at each iteration it selects an edge $e \in E \setminus E'$ that maximizes the potential difference

$$\Phi(E' \cup \{e\}) - \Phi(E').$$

The algorithm terminates when it reaches to a set E with $\Phi(E) = 0$, indicating that the density constraint is satisfied for all input sets $C_i \in \mathcal{C}$. It can be shown that DGreedy provides an approximation guarantee.

Proposition 3.2. DGreedy *is an* $\mathcal{O}(\log \ell)$*-approximation algorithm for the* SparseDens *problem.*

↳ Supplementary material

We obtain Proposition 3.2 by using the classic result of Wolsey [20] on maximizing motonote and submodular functions. The key is to show that the potential function Φ is monotone and submodular.

Increasing

Proposition 3.3. *The potential function* Φ *is monotone and submodular.* "Diminishing Returns"

Communities
Edges $\Big[\quad\Big]_{m\times L}$

Running-time analysis. Let $L = \sum_{i=1}^{\ell} |C_i|$ and $m = |E|$. Consider an $m \times L$ table T so that $T[e,i] = 1$ if $e \in E(C_i)$ and $T[e,i] = 0$ otherwise. It is easy to see that DGreedy can be implemented with a constant number of passes for each non-zero entry of T, giving a running time of $\mathcal{O}(L+m\ell)$. If we use a sparse implementation for T, the overall running time becomes $\mathcal{O}(L + |T|)$, where $|T|$ is the number of non-zero entries of T.

Adding connectivity constraints. Note that solutions to the SparseDens problem may be sparse networks in which communities C_i are dense but *disconnected*. For certain applications we may want do introduce an additional connectivity constraint, so that all subgraphs $G(C_i,E(C_i))$ are connected.

Combining the two constraints of density and connectivity can be handled by a simple adaptation of the greedy algorithm. In particular we can use a new potential that is the sum of the density potential in Equation (3.1) and a potential for connected components. This new potential is still monotone and submodular, thus, a modified greedy will return a solution that satisfies both density and connectivity constraints and provides an $\mathcal{O}(\log \ell)$-approximation guarantee.

4 Sparsification with star constraints

In the second instantiation of the NetSparse problem, the goal is to find a sparse network G' so that each input community $C_i \in \mathcal{C}$ *contains a star*, meaning that for every community C_i subgraph $G(C_i,E(C_i))$ has a star spanning subgraph. The motivation is that a star has small diameter, as well as a central vertex that can act as a community leader. Thus, star-shaped groups have low communication cost and good coordination properties.

We define the SparseStars problem as the special case of Problem 1 by taking ρ to be the star-containment property. We can again show that SparseStars is a computationally hard problem.

Proposition 4.1. *The* SparseStars *problem is* **NP***-hard.* → Supplementary Material

Unfortunately, SparseStars is not amenable to the same approach we used for SparseDens, hence, we use a completely different set of algorithmic techniques. Our algorithm, called `DS2S` (for *"directed stars to stars"*), is based on solving a *directed version* of SparseStars, which is formally defined as follows.

Problem 2. (SparseDiStars) *Consider a directed network $N = (V,A)$, and a set of ℓ communities $\mathcal{C} = \{C_1, \ldots, C_\ell\}$. We want to find a sparse directed network $N' = (V, A')$ with $A' \subseteq A$, such that, the number of edges $|A'|$ is minimized and for each community $C_i \in \mathcal{C}$ there is a central vertex $c_i \in C_i$ with $\boxed{(c_i \to x) \in A'}$ for all $x \in C_i \setminus \{c_i\}$.*

↳ $c_i \to x$ (Dir. edge)

In Problem 2, the original network N and the sparsifier N' are both directed. Note, however, that SparseDiStars can be also defined with an undirected network G as input: just create a directed version of G, by considering each edge in both directions. Thus we can consider that SparseStars and SparseDiStars take the same input. In this case, it is easy to verify the following observation.

Observation 1. *If $G^* = (V,E^*)$ is the optimal solution for* SparseStars, *and $N^* = (V,D^*)$ is the optimal solution for* SparseDiStars, *for the same input, then*

$$|E^*| \leq |D^*| \leq 2|E^*|. \quad \text{[2-approx]}$$

As with SparseStars, the SparseDiStars problem is **NP**-hard.[1] Our approach is to solve SparseDiStars and use the directed sparsifier $N' = (V,D')$ to obtain a solution for SparseStars. Observation 1 guarantees that the solution we obtain for SparseStars in this way is not far from the optimal. In the next paragraph, we describe how to obtain an approximation algorithm for the SparseDiStars problem.

Solving SparseDiStars. First, we observe that SparseDiStars can be viewed as a HyperedgeMatching problem, which is defined as follows: we are given a set of elements X, a collection of *hyperedges* $\mathcal{H} = \{H_1, \ldots H_t\}$, $H_i \subseteq X$, and a score function $c : \mathcal{H} \to \mathbb{R}$. We seek to find a set of disjoint hyperedges $\mathcal{I} \subseteq \mathcal{H}$ maximizing the total score $\sum_{H_i \in \mathcal{I}} c(H_i)$.

Hyperedges
↓
Elements in
Power(X)

The mapping from SparseDiStars to HyperedgeMatching is done as follows: We set X to $\mathcal{C}$. Given a subset $H \subseteq \mathcal{C}$, let us define $\mathsf{I}(H)$ and $\mathsf{U}(H)$ to be the intersection and union of members in H, respectively. Let us construct a set of hyperedges as

$$\mathcal{H} = \{\underbrace{H \mid H \subseteq \mathcal{C} \text{ and } \mathsf{I}(H) \neq \emptyset}_{\text{Partitioning of } \mathcal{C}}\},$$

and assign scores

$$c(H) = 1 - |H| + \sum_{v \in \mathsf{U}(H)} |\{i \mid v \in C_i \in H\}| - 1. \quad \text{(Weird)}$$

Note that if represented naïvely, the resulting hypergraph can be of exponential size. However, this problem can be avoided easily by an implicit representation of the hypergraph. We can now show that the optimal solution to the transformed instance of HyperedgeMatching can be used to obtain an optimal solution to SparseDiStars.

Proposition 4.2. *Let $\mathcal{I}$ be the optimal solution for the* HyperedgeMatching *problem instance. Let $D_\mathcal{I}$ be the union of directed stars, each star having a center in $\mathsf{I}(H)$ and directed edges towards vertices in $\mathsf{U}(H)$ for every $H \in \mathcal{I}$. If D^* is the optimal solution to the* SparseDiStars *problem, then*

$$|D^*| = |D_\mathcal{I}| = \sum_{C_i \in \mathcal{C}} (|C_i| - 1) - \sum_{H \in \mathcal{I}} c(H).$$

Consider a greedy algorithm, `HGreedy`, which constructs a solution to HyperedgeMatching by iteratively adding hyperedges to $\mathcal{J}$ so that it maximizes $\sum_{H_i \in \mathcal{J}} c(H_i)$, while keeping $\mathcal{J}$ disjoint. As shown in the supplementary material, `HGreedy` is a k-factor approximation algorithm for the HyperedgeMatching problem. The proof is based on the concept of k-extensible systems [16].

Proposition 4.3. *Let $\mathcal{J}$ be the resulting set of hyperedges given by the* `HGreedy` *algorithm, and let $\mathcal{I}$ be the optimal solution for* HyperedgeMatching*. Then,*

$$\sum_{H \in \mathcal{I}} c(H) \leq k \sum_{H \in \mathcal{J}} c(H), \quad \text{where} \quad k = \max_{H \in \mathcal{H}} |H|.$$

[k-approx]

Propositions 4.2 and 4.3 imply the following.

Corollary 4.1. *Let D^* be the optimal solution of the* SparseDiStars *problem. Let $\mathcal{J}$ be the greedy solution to the corresponding* HyperedgeMatching *problem, and let $D_\mathcal{J}$ be the corresponding edges (obtained as described in Proposition 4.2). Then,*

$$|D_\mathcal{J}| \leq \frac{k-1}{k} C + \frac{1}{k}|D^*|, \quad \text{where} \quad C = \sum_{C_i \in \mathcal{C}} (|C_i| - 1)$$

and k is the maximum number of sets in $\mathcal{C}$ that have a non-empty intersection.

[1] The proof of Proposition 4.1 can be modified slightly to show **NP**-hardness for SparseDiStars.

Algorithm 1: The DS2S algorithm for SparseStars.

```
    Input: $G_0 = (V,E_0)$ and $\mathcal{C} = \{C_1, \ldots, C_\ell\}$
    Output: Graph $G = (V,E)$ such that $E(C_i)$
            contains a star, for all $i \in 1, \ldots, \ell$.
    $\mathcal{J}$ = HGreedy($\mathcal{C}$);
    $D_{\mathcal{J}}$ = H2D($\mathcal{J}$);
    $E$ = D2E($D_{\mathcal{J}}$);
    return $G = (V,E)$;
```

Putting the pieces together. The pseudo-code of DS2S is shown in Algorithm 1. In the first step, the algorithm invokes HGreedy and obtains a solution to the HyperedgeMatching problem. This solution is then translated into a solution to the SparseDiStars problem (function H2D). Finally, the solution to SparseDiStars is translated into a solution to the SparseStars by transforming each directed edge in $D_{\mathcal{J}}$ into an undirected edge and removing duplicates (function D2E). We have the following result.

Proposition 4.4. *Let E^* be the optimal solution of the SparseStars problem. Let E be the output of the* DS2S *algorithm. Then,*

$$|E| \leq \frac{k-1}{k} C + \frac{2}{k} |E^*|, \quad \textit{where} \quad C = \sum_{C_i \in \mathcal{C}} (|C_i| - 1)$$

and k is the maximum number of sets in $\mathcal{C}$ that have a non-empty intersection.

Running time. The running time of the DS2S algorithm is dominated by HGreedy. The other two steps (lines 2 and 3 in Algorithm 1) require linear time with respect to $|V|$. HGreedy can be implemented with a priority queue. In each step we need to extract the maximum-weight hyperedge, and update all intersecting hyperedges by removing any common sets. The number of maximal hyperedges in $\mathcal{H}$ is at most $|V|$ (one for each vertex v), and assuming that the maximum number of intersecting hyperedges is bounded by c, the total running time of the algorithm is $\mathcal{O}(|V| \ell \log |V| + \ell \sum |C_i|)$.

5 Experimental evaluation

In this section we discuss the empirical performance of our methods. Our experimental study is guided by the following questions.

Q1. How do our algorithms compare against competitive sparsification baselines that also aim at preserving the community structure?

Q2. How well is the structure of the sparsified network preserved compared to the structure of the original network?

Q3. What are specific case studies that support the motivation of our problem formulation?

We note that the implementation of the algorithms and all datasets used will be made publicly available.

Datasets. We use 13 datasets (*D1–D13*); each dataset consists of a network $G = (V,E)$ and a set of communities $\mathcal{C}$. We describe these datasets below, while their basic characteristics are shown in Table 1.

- *KDD* and *ICDM* are subgraphs of the DBLP co-authorship network. Edges represent co-authorships between authors. Communities are formed by keywords that appear in paper abstracts.
- *FB-circles* and *FB-features* are Facebook ego-networks available at the SNAP repository.[2] In *FB-circles* the communities are social-circles of users. In *FB-features* communities are formed by user profile features.
- *lastFM-artists* and *lastFM-tags* are friendship networks of last.fm users.[3] A community in *lastFM-artists* and *lastFM-tags* is formed by users who listen to the same artist and genre, respectively.
- *DB-bookmarks* and *DB-tags* are friendship networks of Delicious users.[4] A community in *DB-bookmarks* and *DB-tags* is formed by users who use the same bookmark and keyword, respectively.

Additionally, we use SNAP datasets with ground-truth communities. To have more focused groups, we only keep communities with size less than 10. To avoid having disjoint communities, we start from a small number of seed communities and iteratively add other communities that intersect at least one of the already selected. We stop when the number of vertices reaches 10 K. In this way we construct the following datasets:

- *Amazon*: Edges in this network represent pairs of frequently co-purchased products. Communities represent product categories as provided by Amazon.
- *DBLP*: This is also a co-authorship network. Communities are defined by publication venues.
- *Youtube*: This is a social network of Youtube users. Communities consist of user groups created by users.

For the case studies we use the following datasets.

- *Cocktails*:[5] Vertices represent drink ingredients and communities correspond to ingredients appearing in cocktail recipes. The *Cocktails* dataset does not have a ground-truth network.
- *Birds*: This dataset consists of group sightings of *Parus Major* (great tit) [8]. The dataset also contains gender, age, and immigrant status of individual birds.

[2] snap.stanford.edu/data/egonets-Facebook.html

[3] grouplens.org/datasets/hetrec-2011/

[4] www.delicious.com

[5] www-kd.iai.uni-bonn.de/InVis

Table 1: Network characteristics. $|V|$: number of nodes; $|E|$: number of edges in the underlying network; $|E_0|$: the number of edges induced by communities; C: the number of connected components; ℓ: number of sets (communities); $\text{avg}(\alpha_0)$: average density of the ground-truth subgraphs induced by the communities; $s_{\min}$, s_{avg}: minimum and average set size; $t_{\max}$, t_{avg}: maximum and average participation of a vertex to a set.

Dataset	$\|V\|$	$\|E\|$	$\|E_0\|$	C	ℓ	$\text{avg}(\alpha_0)$	$s_{\min}$	s_{avg}	$t_{\max}$	t_{avg}
Amazon (*D1*)	10001	25129	17735	7	11390	0.769	2	3.52	20	4.01
DBLP (*D2*)	10001	27687	22264	1	1767	0.581	6	7.46	10	1.31
Youtube (*D3*)	10002	72215	15445	1	5323	0.698	2	4.02	82	2.14
KDD (*D4*)	2891	11208	5521	58	8103	0.178	2	31.16	1288	137.00
ICDM (*D5*)	3140	10689	5079	112	8623	0.183	2	32.46	1339	139.10
FB-circles (*D6*)	4039	88234	55896	1	191	0.640	2	23.15	44	1.53
FB-features (*D7*)	4039	88234	84789	1	1245	0.557	2	29.78	37	9.21
lastFM-artists (*D8*)	1892	12717	5253	20	7968	0.047	2	8.29	1147	36.73
lastFM-tags (*D9*)	1892	12717	7390	20	2064	0.053	2	13.60	50	15.43
DB-bookmarks (*D10*)	1861	7664	1213	62	8337	0.069	2	3.34	58	15.32
DB-tags (*D11*)	1861	7664	6293	62	14539	0.032	2	13.79	658	107.60
Birds (*D12*)	1052	44812	44812	1	49578	1.0	2	6.03	938	284.30
Cocktails (*D13*)	334	3619	3619	1	1600	1.0	2	3.73	427	17.89

Experimental setup. All datasets consist of a graph $G = (V,E)$ and a set of communities $\mathcal{C} = \{C_1, \ldots, C_\ell\}$. The output of our algorithms is a sparsified graph $G^* = (V, E^*)$. Clearly, any reasonable sparsification algorithm would include in E^* only edges that belong in at least one of the graphs $G[C_i] = (C_i, E[C_i])$. Accordingly, we define E_0 to be the set of edges belonging in at least one such subgraph: $E_0 = \cup_{i=1\ldots\ell} E[C_i]$.

SparseDens requires a density threshold α_i for each community $C_i \in \mathcal{C}$. We set this parameter proportional to the density D_i of $G[C_i]$. We experiment with $\alpha_i = \epsilon\, D_i$, for $\epsilon = 0.5$, 0.7, and 0.9.

SparseStars aims to find a star in every community $G[C_i]$. If no star is contained in $G[C_i]$ then the community C_i is discarded.

Baseline. We compare our algorithms with a sparsification method, proposed by Satuluri et al. [18] to enhance community detection, and shown in a recent study by Lindner et al. [14] to outperform its competitors and to preserve well network cohesion. The algorithm, which we denote by `LS`, considers local similarity of vertex neighborhoods. The reader should keep in mind that `LS` is not optimized for the problems we define in this paper, and in fact, it does not use the communities $\mathcal{C}$ as input. Nonetheless we present `LS` in our evaluation, as it is a state-of-the-art sparsification method that aims to preserve community structure.

Amount of sparsification. Starting with input network $G_0 = (V, E_0)$ and communities $\mathcal{C}$ we compute a sparsified network G^*, for datasets *D1*–*D11*. We solve the SparseStars problem using the `DS2S` and the SparseDens problem using the `DGreedy` algorithm for the three different values of ϵ we consider. We quantify the amount of sparsification by the ratio $\rho = |E^*|/|E_0|$, which takes values from 0 to 1, and the smaller the value of ρ the larger the amount of quantification. Boxplots with the distribution of the values of ρ for the four different cases (SparseStars, and SparseDens with $\epsilon = 0.5, 0.7, 0.9$) are shown in Figure 1.

As $\alpha_i \nearrow$, $\rho \nearrow$, Smaller the extent of sparsification.

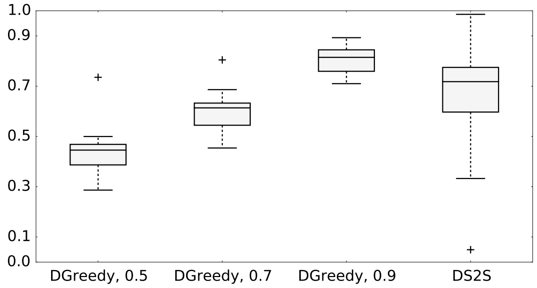

Figure 1: Amount of sparsification shown as distribution of the values $\rho = |E^*|/|E_0|$ over the different datasets for each problem setting.

The `LS` baseline takes as parameter the number of edges in the sparsified network. Thus, for each problem instance we ensure that the sparsified network obtained by `LS` have the same number of edges (up to a 0.05 error margin controlled by `LS`) as the sparsified networks obtained by our methods in the corresponding instances.

Properties of sparsified networks. We start by considering our first two evaluation questions **Q1** and **Q2**. For this, we compare our methods with a competitive

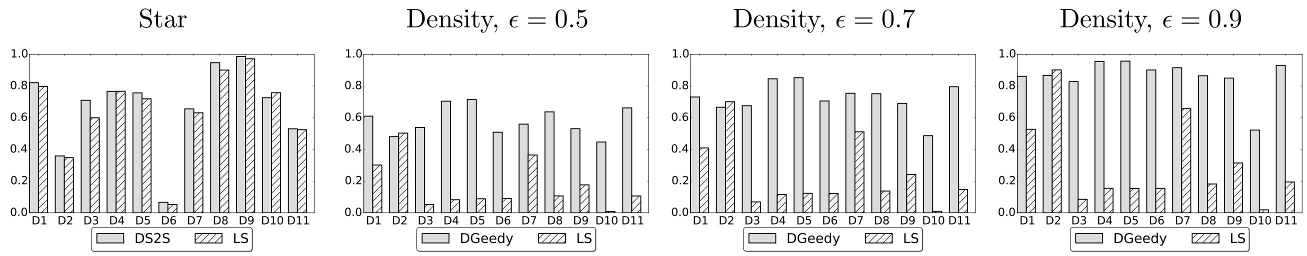

Figure 2: Relative degree δ within communities, on the sparsified graphs G^* produced by DGreedy and baseline LS for datasets *D1–D11*. Measure δ is defined as average degree within community in the sparsified graph divided by average degree within community in the input graph. Larger values of δ correspond to better preserved community sets.

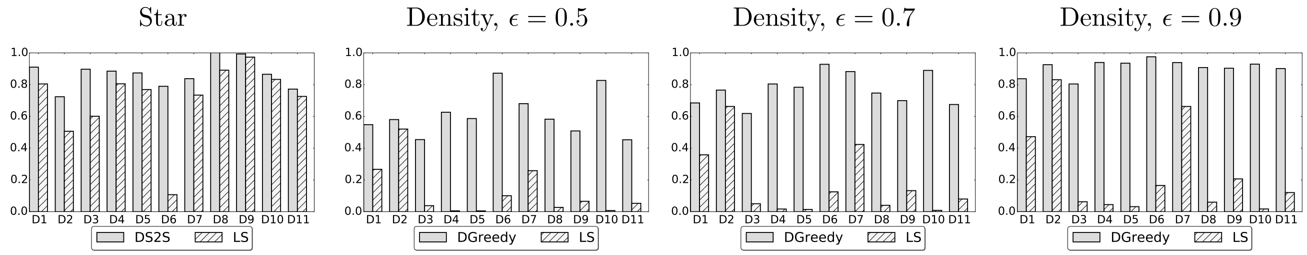

Figure 3: Relative paths length λ, on the sparsified graphs G^* produced by DGreedy and baseline LS for datasets *D1–D11*. Measure λ is defined as harmonic mean of within communities shortest-path lengths in the input graph divided by corresponding harmonic mean in the sparsified graph. Again, larger values of λ correspond to better preserved community sets.

baseline (LS), and we quantify the amount structure preservation in the sparsified network.

Recall that for an input network $G_0 = (V, E_0)$ and communities $\mathcal{C}$ we compute a sparsified network G^*, for datasets *D1–D11*. We *compare* the networks G^* and G_0 by computing the *average degree* and the *average shortest-path length* within the communities $\mathcal{C}$.[6] The goal is to test whether within-communities statistics in the sparsified network are close to those in the original network. The results for average degree and average shortest path are shown in Figures 2 and 3, respectively. The leftmost panel in each figure shows the results for the the SparseStars problem, while the other three panels show the results for the SparseDens problem, for the three different values of ϵ we consider.

As expected, in the sparsified network, average degrees decrease and short-path lengths increase. For SparseDens, as ϵ increases, both average distance and average shortest-path length in the sparsified network come closer to their counterparts in the input network. For the SparseStars problem the LS baseline is competitive and in most cases it produces networks whose statistics are close to the ones of the networks produced by DS2S. However, for the SparseStars problem, the LS baseline does not do a particularly good job in preserving community structure.

Overall this experiment reinforces our understanding that while sparsification is effective with respect to reducing the number of edges, the properties of the communities in the sparsified network resemble respective properties in the input network.

Running time. For all reported datasets the total running time of DS2S is under 1 second, while DGreedy completes in under 5 minutes. The experiments are conducted on a machine with Intel Xeon 3.30GHz and 15.6GiB of memory.

Case studies. To address evaluation question **Q3** we conduct two case studies, one presented here and one in the supplementary material. In both cases there is no underlying network, so they can be considered instances of the *network design* problem.

[6] The average shortest-path length is estimated using the harmonic mean, which is robust to disconnected components.

Table 2: Top-10 star centers, discovered by DS2S algorithm on *Cocktails* dataset. The centers are ordered by the discovered order, with the number of sets a center covers in parentheses.

vodka (202)	gin (86)
orange j. (118)	amaretto (85)
pineapple j. (86)	light rum (58)
bailey's (78)	kahlua (58)
tequila (81)	blue curacao (50)

Cocktails case study. In this case the input communities are defined by the ingredients of each cocktail recipe. We first run the DS2S algorithm on the input sets, and we obtain network G^* with 1 593 edges, that is, around 44 % of the edges of G, giving an average degree of 9.5. The first ten star centers, in the order selected by the DS2S algorithm are shown in Table 2. The table also shows the number of cocktails that each ingredient serves as a star. We see that the algorithm selects popular ingredients that form the basis for many cocktails. A snippet of the reconstructed network is shown in Figure 4.

In order to compare the outputs of DS2S (G_1^*) and DGreedy, we ran the latter with density parameter $\alpha = 0.65$. For this value of α we get G_2^* having 1 420 edges, so that we can have a more meaningful comparison of G_1^* and G_2^*. In Figure 5 we depict the degree of each vertex in the two reconstructed networks, G_1^* and G_2^*, as a function of their degree in the underlying network G. From the figure, we observe that *some* of the unusually high-degree vertices in G maintain their high degree in G_1^*; these are probably the vertices that the DS2S algorithm decides that they serve as star centers. On the other hand, there are other high-degree vertices in G that lose their high-degree status in G_1^*; these are the vertices that the DS2S algorithm did not use as star centers. On the other hand, the DGreedy algorithm sparsifies the feasibility network much more uniformly and vertices maintain their relative degree in G_2^*.

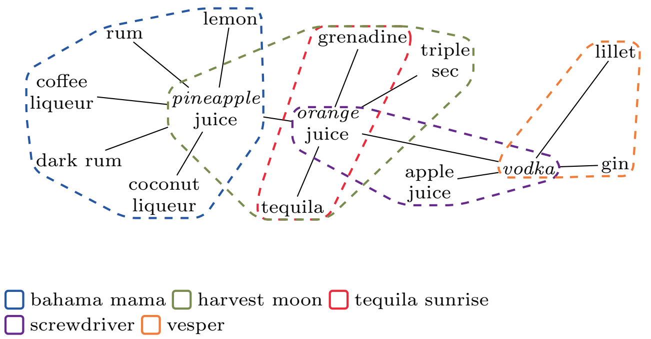

Figure 4: A snippet of the discovered network for the *Cocktails* dataset.

Base Degree = reconstructed degree

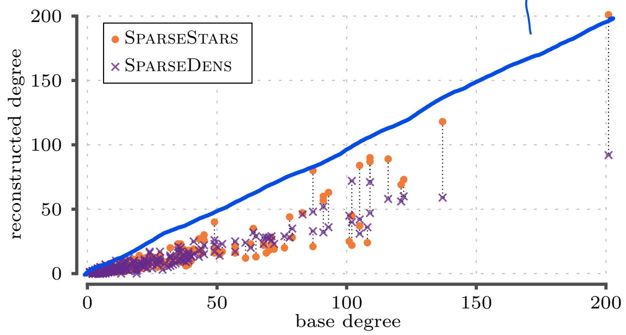

Figure 5: Vertex degree of reconstructed networks as a function of the vertex degree in the base network in the *Cocktails* dataset.

6 Related work

To the best of our knowledge, we are the first to introduce and study the SparseDens and SparseStars problems. As a result, the problem definitions, technical results, and algorithms presented in this paper are novel. However, our problems are clearly related to *network sparsification* and *network design* problems.

Network sparsification: Existing work on network sparsification aims to simplify the input network — by removing edges — such that the remaining network maintains some of the properties of the original network. Such properties include shortest paths and connectivity [7, 21], cuts [1, 9], source-to-sink flow [17], spectral properties [2, 6, 19], modularity [5], as well as information-propagation pathways [15]. Other work focuses on sparsification that improves network visualization [14]. The main difference of our paper and existing work is that we consider sparsifying the network while maintaining the structure of a given set of communities. Such community-aware sparsification raises new computational challenges that are distinct from the computational problems studied in the past.

Network design problems: At a high level, network-design problems consider a set of constraints and ask to construct a minimum-cost network that satisfies those constraints [3, 4, 11–13]. As in our case, cost is usually formulated in terms of the number of edges, or total edge weight. Many different constraints have been considered in the literature: reachability, connectivity, cuts, flows, etc. Among the existing work in network design, the most related to our paper is the work by Angluin et al. [4] and by Korach and Stern [12, 13]. Using the notation introduced in Section 2, Angluin et al. essentially solve the NetSparse problem with the $\mathbb{I}_c$ property. Our results on the SparseDens problem and its variant on connected SparseDens are largely inspired by the work of Angluin et al. On the other hand, for the SparseStars problem we need to introduce completely new techniques, as the submodularity property is not applicable in this case. Korach and Stern [12, 13] study the following problem: given a collection of sets, construct a minimum-weight *tree* so that each given set defines a star in this tree. Clearly, this problem is related to the SparseStars problem considered here, however, the tree requirement create a very significant differentiation: the problem studied by Korach and Stern is polynomially-time solvable, while SparseStars is **NP**-hard. In terms of real-world applications, while tree structures are well motivated in certain cases (e.g., overlay networks), they are not natural in many other (e.g., social networks).

7 Concluding remarks

In this paper, we have introduced NetSparse, a new formulation of network sparsifcation, where the input consists not only of a network but also of a set of communities. The goal in NetSparse is twofold: (*i*) sparsify the input network as much as possible, and (*ii*) guarantee some connectedness property for the subgraphs induced by the input communities on the sparsifiers. We studied two connectedness properties and showed that the corresponding instances of NetSparse is **NP**-hard. We then designed effective approximation algorithms for both problems. Our experiments with real datasets obtained from diverse domains, verified the effectiveness of the proposed algorithms, in terms of the number of edges they removed. They also demonstrated that the obtained sparsified networks provide interesting insights about the structure of the original network with respect to the input communities.

Acknowledgements. This work was funded by Tekes project "Re:Know," Academy of Finland project "Nestor" (286211), EC H2020 RIA project "SoBigData" (654024), a Microsoft gift, a Nokia Faculty Fellowship, and by NSF grants: IIS 1320542, IIS 1421759 and CAREER 1253393.

References

[1] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *PODS*, 2012.

[2] K. J. Ahn, S. Guha, and A. McGregor. Spectral sparsification in dynamic graph streams. In *RANDOM-APPROX*, pages 1–10, 2013.

[3] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, and J. Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms*, 2(4):640–660, 2006.

[4] D. Angluin, J. Aspnes, and L. Reyzin. Network construction with subgraph connectivity constraints. *J. of Comb. Optimization*, 2013.

CONNECTIVITY as a property for community construction

[5] A. Arenas, J. Duch, A. Fernández, and S. Gómez. Size reduction of complex networks preserving modularity. *New Journal of Physics*, 9(6):176, 2007.

[6] J. D. Batson, D. A. Spielman, N. Srivastava, and S. Teng. Spectral sparsification of graphs: theory and algorithms. *CACM*, 56(8):87–94, 2013.

[7] M. Elkin and D. Peleg. Approximating k-spanner problems for $k > 2$. *Theoretical Computer Science*, 337(1):249–277, 2005.

[8] D. Farine. The role of social and ecological processes in structuring animal populations. *Royal Society Open Science*, 2(4), 2015.

[9] W. S. Fung, R. Hariharan, N. J. Harvey, and D. Panigrahi. A general framework for graph sparsification. In *STOC*, 2011.

[10] M. Garey and D. Johnson. *Computers and intractability: a guide to the theory of NP-completeness.* WH Freeman & Co., 1979.

[11] A. Gupta, R. Krishnaswamy, and R. Ravi. Online and stochastic survivable network design. *SIAM Journal of Computing*, 41(6):1649–1672, 2012.

[12] E. Korach and M. Stern. The clustering matroid and the optimal clustering tree. *Math. Program.*, 98(1-3):385–414, 2003.

[13] E. Korach and M. Stern. The complete optimal stars-clustering-tree problem. *Discrete Applied Mathematics*, 156(4):444–450, 2008.

[14] G. Lindner, C. L. Staudt, M. Hamann, H. Meyerhenke, and D. Wagner. Structure-preserving sparsification of social networks. In *ASONAM*, 2015.

→ Baseline

[15] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsification of influence networks. In *KDD*, 2011.

[16] J. Mestre. Greedy in approximation algorithms. In *ESA*, 2006.

[17] E. Misiolek and D. Z. Chen. Two flow network simplification algorithms. *IPL*, 97(5):197–202, 2006.

[18] V. Satuluri, S. Parthasarathy, and Y. Ruan. Local graph sparsification for scalable clustering. In *SIGMOD*, 2011.

→ Baseline

[19] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.

Recommended by Prof

[20] L. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393, 1982.

[21] F. Zhou, S. Mahler, and H. Toivonen. Network simplification with minimal loss of connectivity. In *ICDM*, 2010.

Supplementary material

Community-aware network sparsification

Aristides Gionis
Polina Rozenshtein
Nikolaj Tatti
Evimaria Terzi

S1. Proofs

Proof. [Proof of Proposition 3.1] We consider the decision version of SPARSEDENS. The problem is obviously in **NP**. To prove the hardness we provide a reduction from from the HITTINGSET problem. An instance of HITTINGSET consists of a universe of items U, a collection of sets $C_i \subseteq U$, and a positive integer c. The task is to decide whether there exists a "hitting set" $X \subseteq U$, of cardinality at most c, such that $C_i \cap X \neq \emptyset$, for every i.

Consider an instance of the HITTINGSET problem. We will show how to obtain a solution for this instance, using SPARSEDENS. We proceed as follows. First we create a graph $G_0 = (V,E_0)$, such that $|V| = |U| + 1$: for every item $u \in U$ we create a vertex $x_u \in V$, in addition we add one special vertex $s \in V$. The graph is fully connected, $|E_0| = \binom{|V|}{2}$.

Now for every set C_i in the instance of HITTINGSET we create a set of vertices S_i for our problem such that $S_i = \{s\} \cup \{x_u \mid u \in C_i\}$. We also set $\alpha_i = \alpha = \binom{|V|}{2}^{-1}$. Note that α is so low that to satisfy the constraint $\mathbb{I}_{d \geq \alpha}(G,S_i)$ it is sufficient to have $E(S_i) \geq 1$.

Let $G = (V,E)$ be a solution for SPARSEDENS, if one exists. We can safely assume that each edge in E is adjacent to s. To see this, assume that $e = (x_u, x_v) \in E$ and modify E by adding (x_u, s), if not already in E, and deleting e. By doing this swap, we do not violate any constraint since any S_i that contains e will also contain (x_u, s) and having one edge is enough to satisfy the constraint. Moreover, we do not increase the number of edges in E.

Using this construction, we can see that the adjacent vertices in E, excluding s, correspond to the hitting set; that is, there exists a solution to the HITTINGSET problem of cardinality at most c if and only if there exists a solution to SPARSEDENS that uses at most c edges. □

It is interesting to observe that our proof implies that the SPARSEDENS problem is **NP**-hard even if the feasibility network G_0 is fully-connected.

Proof. [Proof of Proposition 3.3] Showing that Φ is monotone is quite straightforward, so we focus on submodularity. We need to show that for any set of edges $X \subseteq Y \subseteq E_0$ and any edge $e \notin Y$ it is

$$\Phi(Y \cup \{e\}) - \Phi(Y) \leq \Phi(X \cup \{e\}) - \Phi(X).$$

Since Φ is a summation of terms, as per Equation (3.2), it is sufficient to show that each individual term is submodular. Thus, we need to show that for any $S_i \in \mathcal{S}$, $X \subseteq Y \subseteq E_0$, and $e \notin Y$ it is

$$\Phi(Y \cup \{e\},S_i) - \Phi(Y,S_i) \leq \Phi(X \cup \{e\},S_i) - \Phi(X,S_i).$$

To show the latter inequality, first observe that for any $S_i, Z \subseteq E_0$ and $e \notin Z$ the difference $\Phi(Z \cup \{e\},S_i) - \Phi(Z,S_i)$ is either 0 or 1. Now fix S_i, $X \subseteq Y \subseteq E_0$, and $e \notin Y$; if $\Phi(X \cup \{e\},S_i) - \Phi(X,S_i) = 0$, either the set of edges X satisfy the density constraint on S_i, or e is not incident in a pair of vertices in S_i. In the latter case, $\Phi(Y \cup \{e\},S_i) - \Phi(Y,S_i) = 0$, as well. In the former case, if X satisfies the density constraint, since $X \subseteq Y$, then the set of edges Y should also satisfy the density constraint, and thus $\Phi(Y \cup \{e\},S_i) - \Phi(Y,S_i) = 0$. □

Proof. [Proof of Proposition 4.1] We consider the decision version of the SPARSESTARS problem. Clearly the problem is in **NP**. To prove the completeness we will obtain a reduction from the 3D-MATCHING problem, the 3-dimensional complete matching problem [10]. An instance of 3D-MATCHING consists of three disjoint finite sets X, Y, and Z, having the same size, and a collection of m sets $\mathcal{C} = \{C_1, \ldots, C_m\}$ containing exactly one item from X, Y, and Z, so that $|C_i| = 3$. The goal is to decide whether there exists a subset of $\mathcal{C}$ where each set is disjoint and all elements in X, Y, and Z are covered.

Assume an instance of 3D-MATCHING. For each C_i create four vertices p_i, u_i, v_i, and w_i. Set the network $G_0 = (V, E_0)$ to be a fully connected graph over all those vertices. Define $P = \{p_i\}$, the set of p_i's. For each $x \in X$, create a set $S_x = \{p_i, u_i, v_i \mid x \in C_i\}$. Similarly, for each $y \in Y$, create a set $S_y = \{p_i, u_i, w_i \mid y \in C_i\}$ and, for each $z \in Z$, create a set $S_z = \{p_i, v_i, w_i \mid z \in C_i\}$. Let $\mathcal{S}$ consist of all these sets.

Let $G = (V,E)$ be the optimal solution to the SPARSESTARS problem; such a solution will consist of induced subgraphs $G_i = (S_i,E(S_i))$ that contain a star. Let μ be the function mapping each S_i to a vertex that acts as a center of the star defined by G_i. Let $O = \{\mu(S_i); S_i \in \mathcal{S}\}$ be the set of these center vertices in the optimal solution. We can safely assume that $O \subseteq P$; even if in the optimal solution there exists an S_i with $E(S_i)$ not intersecting with any other $E(S_j)$, p_i can be picked as the center of this star. For each $o \in O$, define $\mathcal{N}_o = \{S_i \in \mathcal{S} \mid \mu(S_i) = o\}$.

The number of edges $|E|$ in the optimal graph $G = (V,E)$ is equal to $\sum_{S_i \in \mathcal{S}}(|S_i| - 1) - D$, where D is the number the edges that are counted in more than one star. To account for this double counting we proceed as follows: if $\mathcal{N}_o$ contains two sets, then there is one edge adjacent to o that is counted twice. If $\mathcal{N}_o$ contains three sets, then there are three edges adjacent to o that are counted twice. This leads to

$$|E| = \sum_{S_i \in \mathcal{S}} (|S_i| - 1) - \sum_{o \in O} \left(\mathbb{I}_{[|\mathcal{N}_o|=2]} + 3\mathbb{I}_{[|\mathcal{N}_o|=3]} \right),$$

where $\mathbb{I}$ is the indicator function with $\mathbb{I}_{[A]} = 1$ if the statement A is true, and 0 otherwise.

To express the number of edges solely with a sum over the sets, we define a function f as $f(1) = 0$, $f(2) = 1/2$ and $f(3) = 1$. Then

$$|E| = \sum_{S_i \in \mathcal{S}} \left(|S_i| - 1 - f(|\mathcal{N}_{\mu(S_i)}|) \right).$$

Let $\mathcal{Q} \subseteq \mathcal{C}$ be the set 3-dimensional edges corresponding to the set of selected star centers O. Set $t = \sum_{S_i \in \mathcal{S}} (|S| - 2)$. Then $|E| \leq t$ if and only if every $\mathcal{N}_o$ contains 3 sets, which is equivalent to $\mathcal{Q}$ containing disjoint sets that cover X and Y and Z. □

Proof. [Proof of Observation 1] The first part of the inequality follows from the fact that any solution (V,D) for SparseDiStars can be translated to a solution for SparseStars by simply ignoring the edge directions and removing duplicates, if needed. The second part of the inequality follows from the fact that any solution (V,E) for SparseStars can be translated to a feasible solution for SparseDiStars, with at most two times as many edges, by creating two copies each edge (x,y) in E: one for $(x \to y)$ and one for $(y \to x)$. □

To prove Proposition 4.2 we will use the following lemma which we state without the proof.

Lemma 7.1. *Let $H \in \mathcal{H}$ be a hyper-edge and let T be a star with the center x in $\mathsf{I}(H)$ connecting to every vertex in $\mathsf{U}(H) \setminus \{x\}$. The number of edges in T is equal to*

$$\sum_{C \in H} (|C| - 1) - c(H).$$

Now we are ready to prove Proposition 4.2.

Proof. [Proof of Proposition 4.2] Let us first prove $|D^*| \leq |D_{\mathcal{I}}|$. By definition, $\mathcal{H}$ contains all sets C_i as singleton groups. Therefore, each set C_i is included in some $H \in \mathcal{I}$. Hence, $D_{\mathcal{I}}$ is a feasible solution for SparseDiStars and therefore $|D^*| \leq |D_{\mathcal{I}}|$.

We will now prove the other direction. By definition, D^* is a union of stars $\{T_i\}$, where each $T_i = (C_i, A_i)$. Define a family $\mathcal{P}$ by grouping each C_i sharing the same star center. Note that $\mathcal{P}$ is a disjoint subset of $\mathcal{H}$, consequently, it is a feasible solution for HyperedgeMatching. Lemma 7.1 implies that

$$\begin{aligned} |D^*| &= \sum_{H \in \mathcal{P}} \sum_{C \in H} |C| - 1 - c(H) \\ &= \sum_{C \in \mathcal{C}} (|C| - 1) - \sum_{H \in \mathcal{P}} c(H) \\ &\geq \sum_{C \in \mathcal{C}} (|C| - 1) - \sum_{H \in \mathcal{I}} c(H), \end{aligned}$$

where the first equality follows from the fact that the joined trees are edge-disjoint.

Lemma 7.1 implies that

$$|D_{\mathcal{I}}| \leq \sum_{H \in \mathcal{I}} \sum_{C \in H} |C| - 1 - c(H) = \sum_{C \in \mathcal{C}} (|C| - 1) - \sum_{H \in \mathcal{I}} c(H),$$

where the last equality follows since each set C_i is included in some $H \in \mathcal{I}$. □

Proof. [Proof of Proposition 4.3] The set of feasible solutions of the HyperedgeMatching problem forms a k-extensible system [16]. As shown by Mestre [16], the greedy algorithm provides a k-factor approximation to the problem of finding a solution with the maximum weight in a k-extensible system. □

Proof. [Proof of Proposition 4.4] For the solution of the `DS2S` problem we know that $|E| \leq |D_{\mathcal{J}}|$. This is because we can obtain E from $D_{\mathcal{J}}$ by ignoring edge directions, and possibly removing edges, if needed. From the latter inequality, Observation 1, and Corollary 4.1, the statement follows. □

S2. Extension to weighted networks

Our problem definition can be extended for *weighted* graphs $G = (V,E,d)$, where V and E are the sets of nodes and edges in the network. In this case, we assume that edges are weighted by a distance function $d : E \to \mathbb{R}_+$. Small distances indicate strong connections and large distances indicate weak connections. The distance of an edge $e \in E$ is denoted by $d(e)$, while the total distance of a set of edges $E' \subseteq E$ is defined as $d(E') = \sum_{e \in E'} d(e)$. Given such a weighted network, we can extend the definition of the NetSparse problem as follows:

Problem 3. (WeightedNetSparse) *Consider an underlying network $G = (V, E, d)$, where d represent* *edge distances, and let ρ be a graph property. Given a set of ℓ communities $\mathcal{C} = \{C_1, \ldots, C_\ell\}$, we want to construct a* sparse network *$G' = (V, E')$, such that, (i) $E' \subseteq E$; (ii) $\mathbb{I}_\rho(G', C_i) = 1$, for all $C_i \in \mathcal{C}$; and (iii) the sum of distances of edges in the sparse network, $d(E') = \sum_{e \in E'} d(e)$, is minimized.*

As before, depending on whether ρ is the connectivity, the density or the star-containment property, we get the corresponding weighted versions of the SparseConn, SparseDens and SparseStars problems respectively. The greedy algorithms developed for the SparseConn and SparseDens problems can be also used for their weighted counterparts. In particular, in the greedy step of the algorithm the next edge is chosen so as to maximize the potential difference

$$\frac{\Phi(E' \cup \{e\}) - \Phi(E')}{d(e)}.$$

However, the algorithm we give for SparseStars is only applicable to unweighted networks; developing a new algorithm for the weighted case is left as future work.

S3. Birds case study

We present a second case study where the input communities are group sightings of birds. We run the `DS2S` algorithm on the input sets, and we obtain a sparsified network with 809 star centers and 21 077 edges, that is, around 47 % of the edges of input network. The dataset also contains gender (male/female/unknown), age (juvenile/adult), and immigration status of each individual bird. We studied whether some characteristics are favoured when selecting centers. Here, we found out that juveniles are preferred as centers, as well as, male residents, see Figure 6.

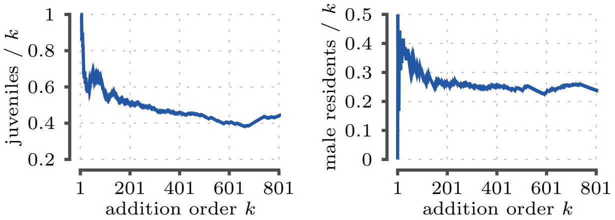

Figure 6: Proportion of juveniles and male residents in top-k selected star centers in *Birds* as a function of k.