# On the Complexity of Community-Aware Network Sparsification

**Emanuel Herrendorf**
Philipps-Universität Marburg, Fachbereich Mathematik und Informatik, Germany

**Christian Komusiewicz**
Friedrich Schiller University Jena, Institute of Computer Science, Germany

**Nils Morawietz**
Friedrich Schiller University Jena, Institute of Computer Science, Germany

**Frank Sommer**
Friedrich Schiller University Jena, Institute of Computer Science, Germany

## Abstract

In the NP-hard $\Pi$-Network Sparsification problem, we are given an edge-weighted graph $G$, a collection $\mathcal{C}$ of $c$ subsets of $V(G)$, called *communities*, and two numbers $\ell$ and $b$, and the question is whether there exists a spanning subgraph $G'$ of $G$ with at most $\ell$ edges of total weight at most $b$ such that $G'[C]$ fulfills $\Pi$ for each community $C \in \mathcal{C}$. We study the fine-grained and parameterized complexity of two special cases of this problem: Connectivity NWS where $\Pi$ is the connectivity property and Stars NWS, where $\Pi$ is the property of having a spanning star.

First, we provide a tight $2^{\Omega(n^2+c)}$-time running time lower bound based on the ETH for both problems, where $n$ is the number of vertices in $G$ even if all communities have size at most 4, $G$ is a clique, and every edge has unit weight. For the connectivity property, the unit weight case with $G$ being a clique is the well-studied problem of computing a hypergraph support with a minimum number of edges. We then study the complexity of both problems parameterized by the feedback edge number $t$ of the solution graph $G'$. For Stars NWS, we present an XP-algorithm for $t$ answering an open question by Korach and Stern [Discret. Appl. Math. '08] who asked for the existence of polynomial-time algorithms for $t = 0$. In contrast, we show for Connectivity NWS that known polynomial-time algorithms for $t = 0$ [Korach and Stern, Math. Program. '03; Klemz et al., SWAT '14] cannot be extended to larger values of $t$ by showing NP-hardness for $t = 1$.

**2012 ACM Subject Classification** Theory of computation → Parameterized complexity and exact algorithms; Mathematics of computing → Hypergraphs

**Keywords and phrases** parameterized complexity, hypergraph support, above guarantee parameterization, exponential-time-hypothesis

**Digital Object Identifier** 10.4230/LIPIcs.MFCS.2024.60

**Related Version** *Continuously Updated Version*: https://arxiv.org/abs/2402.15494

**Funding** *Nils Morawietz*: Partially supported by the DFG, project OPERAH (KO 3669/5-1).
*Frank Sommer*: Partially supported by the DFG, project EAGR (KO 3669/6-1)

**Acknowledgements** Some of the results of this work are also contained in the first author's Masters thesis [23].

## 1 Introduction

A common goal in network analysis is to decrease the size of a given network to speed up downstream analysis algorithms or to decrease the memory footprint of the graphs. This leads to the task of network sparsification where one wants to reduce the number of edges of a network while preserving some important property $\Pi$ [6, 31, 35]. Similarly, in network


49th International Symposium on Mathematical Foundations of Computer Science (MFCS 2024).
Editors: Rastislav Královič and Antonín Kučera; Article No. 60; pp. 60:1–60:18
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

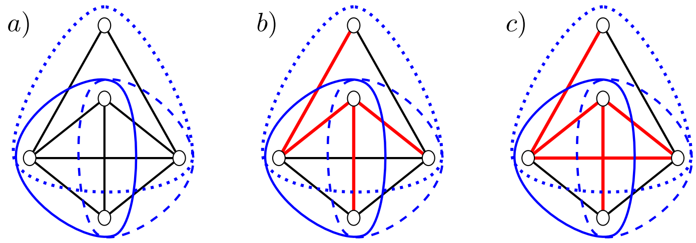


**Figure 1** *a*) The communities (blue) and input graph of a Π-NWS instance. *b*) and *c*) Optimal solutions (red) for Unweighted Connectivity NWS, and Unweighted Stars NWS, respectively.

design the task is often to construct a minimum-size or minimum-weight network fulfilling a given property, the most famous example being Minimum-Weight Spanning Tree.

In many applications the input contains, in addition to a network, a hypergraph on the same vertex set [17, 24, 32]. The hyperedges of this hypergraph represent, for example, communities that are formed within the network. In presence of such community data, the sparsified network should preserve a property not for the whole network but instead for each community, that is, for each hyperedge of the hypergraph. Gionis et al. [19] called this task *community-aware network sparsification* and formalized it as follows.

Π-Network Sparsification (Π-NWS)

**Input:** A graph $G$, a collection $\mathcal{C}$ of $c$ subsets of $V(G)$, called *communities*, an edge-weight function $\omega : E(G) \to \mathbb{R}_+$, an integer $\ell$, and a positive real number $b$.
**Question:** Is there a graph $G' = (V(G), E')$ with $E' \subseteq E(G)$, $|E'| \leq \ell$, and total edge weight at most $b$ such that for each community $C_i \in \mathcal{C}$ the subgraph of $G'$ induced by $C_i$ satisfies Π?

We say that a graph $G'$ fulfilling the requirements is a *solution* for the instance $I$. A very well-studied property Π, considered by Gionis et al. [19] but also in many previous works [2, 7, 13, 15, 28] is that every community should induce a connected subgraph. A graph $G$ that has this property for some hypergraph $H$, is called a *support* for $H$ [4, 5, 28]. We denote the corresponding special case of Π-NWS as Connectivity NWS. Another variant of Π-NWS, also studied by Gionis et al. [19], is to demand that every community not only induces a connected subgraph but more strongly that it contains a *spanning star*. In other words, in the solution graph $G'$, every community must be contained in the neighborhood of at least one of its vertices, called a *center vertex*. We refer to this variant as Stars NWS. An example instance and solutions for both problems are given in Figure 1.

Connectivity NWS and Stars NWS are both NP-hard [13, 10, 9, 19]. Motivated by this, we study both problems in terms of their parameterized and fine-grained complexity. We also investigate the versions of both problems where each edge has unit weight and refer to them as Unweighted Connectivity NWS and Unweighted Stars NWS.

Our two main results are as follows:

- We show that, based on the Exponential Time Hypothesis (ETH), Connectivity NWS and Stars NWS do not admit algorithms with running time $2^{o(n^2)+c}$, even if the input graph is a clique with unit weights and each community has size at most 4. This bound is matched by simple brute-force algorithms.
- We show that Stars NWS admits an XP-algorithm when parameterized by $t$, the feedback edge number of the solution graph. This positively answers the question of Korach and Stern [30] who asked whether there is a polynomial-time algorithm for finding an optimal solution for Stars NWS that is a tree. In fact, our algorithm extends the polynomial-time solvable cases to solutions that are tree-like.

We obtain several further results, for example a complexity dichotomy for Stars NWS and Unweighted Stars NWS parameterized by $c$, the number of communities.

**Known results.** Already the most basic variant of Connectivity NWS, where the edges have unit weights and the input graph $G$ is a clique, appears in many applications, ranging from explanation of protein complexes [32] to combinatorial auctions [10] to the construction of P2P overlay networks in publish/subscribe systems [8, 24]. Thus, the problem has been studied intensively under various names [2, 7, 8, 13, 15, 24] from a parameterized complexity [7, 15, 24] and an approximation algorithms [2, 8, 24] perspective. For example, the problem is NP-hard even for instances with maximum community size 3 [13], and admits FPT-algorithms for the number of communities and for the largest community size plus the feedback edge number $t$ of a solution [7]. A particular restriction of the problem is to determine whether there is an acyclic solution, called *tree support* or *clustered spanning tree*. It can be determined in polynomial time whether a hypergraph has a tree support and different polynomial-time algorithms have been described over the years [3, 10, 14, 16, 20, 27, 33, 34].

Unweighted Connectivity NWS with general input graphs $G$, has applications in the context of placing green bridges [17, 22]. Unweighted Connectivity NWS is NP-hard even when the maximum degree of $G$ is 3 [22] and even for seven communities [17]. On the positive side, one can construct in polynomial time a tree support if one exists [21, 28, 29].

For Connectivity NWS where we may have arbitrary edge-weights, the distinction whether or not $G$ is restricted to be a clique vanishes: any non-clique input graph $G$ may be transformed into a clique by adding the missing edges with a prohibitively large edge weight. The problem of finding a minimum-weight tree support received attention due to its applications in network visualization [28]. As shown by Korach and Stern [29] and Klemz et al. [28], one can compute minimum-weight tree supports in polynomial time. Gioinis et al. [19] provided approximation algorithms for the general problem.

Stars NWS has received less attention than Connectivity NWS. Gionis et al. [19] showed NP-hardness and provided approximation algorithms. Korach and Stern [30] studied a variant of Stars NWS where the input graph is a clique and the solution is constrained to be a tree $T$ where the closed neighborhood of the center vertex of a community $C_i$ is exactly the community $C_i$. This implies that two different communities need to have different center vertices and thus restricts the allowed set of solution graphs strictly compared to Stars NWS. Korach and Stern [30] showed that this problem is solvable in polynomial time. As an open question, they ask whether this positive result can be lifted to Stars NWS.

Cohen et al. [9] studied the Minimum $\mathcal{F}$-Overlay problem which can be viewed as the following special case of $\Pi$-NWS: The input graph $G$ is a clique and all edges have unit weight; $\mathcal{F}$ is a family of graphs and the property $\Pi$ is to have some spanning subgraph which is contained in $\mathcal{F}$. Unweighted Connectivity NWS and Unweighted Stars NWS with clique input graphs are special cases of Minimum $\mathcal{F}$-Overlay. Cohen et al. [9] provide a complexity dichotomy with respect to properties of $\mathcal{F}$. For most cases of $\mathcal{F}$, Minimum $\mathcal{F}$-Overlay is NP-hard. In particular, Unweighted Stars NWS is NP-hard even when $G$ is a clique [9]. Gionis et al. [19] also studied $\Pi$ being that each community needs to induce a subgraph exceeding some prespecified density. Fluschnik and Kellerhals [17] considered further properties $\Pi$, for example the property of having small diameter.

**Our results and organization of the work.** To put our main results into context, we first summarize in Section 2 some complexity results that follow from simple observations or from previous work. They imply in particular that Stars NWS and Connectivity NWS have an FPT-algorithm for the parameter solution size $\ell$ and that they are W[1]-hard with respect to the dual parameter $k := m - \ell$ even in the unit weight case when $G$ is a clique. Then,

**Table 1** An overview of the parameterized complexity results. A ‡ indicates that this result also holds in the unweighted case and a † indicates that this result only holds in the unweighted case.

| Parameter | Stars NWS | Connectivity NWS |
|---|---|---|
| $\ell$ | FPT (Proposition 2.2, [17]), no polynomial kernel$^\ddagger$ (Proposition 2.4, [17]) | |
| $k := m - \ell$ | W[1]-hard$^\ddagger$ (Proposition 2.3) | |
| $t$ | XP (Theorem 4.1) | P for $t = 0$ ([28])<br>NP-h for $t = 1^\ddagger$ (Theorem 4.9) |
| $c$ | FPT$^\dagger$ (Theorem 5.2)<br>no polynomial kernel$^\ddagger$ (Theorem 5.3)<br>W[1]-h (Theorem 5.1) | NP-h for $c = 7^\ddagger$ ([17]) |
| $\Delta$ | NP-h for $\Delta = 6^\ddagger$ (Corollary 3.3) | NP-h for $\Delta = 3^\ddagger$ ([22]) |

in Section 3 we show that Unweighted Connectivity NWS and Unweighted Stars NWS do not admit algorithms with running time $2^{o(n^2+c)}$ even when $G$ is a clique unless the Exponential Time Hypothesis (ETH) [26] is false.

In Section 4, we consider parameterization by $t$, the feedback edge number of the solution graph $G'$. This is the minimum number of edges that need to be deleted to transform the solution into a forest.[1] The study of $t$ is motivated as follows: The solution size $\ell$ is essentially at least as large as $n - 1$, and thus neither small in practice nor particularly interesting from an algorithmic point of view. Thus, $t$ can be seen as a parameterization above the lower bound $n - 1$. Our first main result is an XP-algorithm for Stars NWS. This positively answers the question of Korach and Stern [30] who asked whether there is a polynomial-time algorithm for $t = 0$ and extends the tractability further to every constant value of $t$. We then show that, in contrast, Unweighted Connectivity NWS is NP-hard already if $t = 1$. Thus, the polynomial-time algorithms for $t = 0$ [29, 28] cannot be lifted to larger values of $t$.

Finally, in Section 5 we study Stars NWS parameterized by the number $c$ of input communities. We obtain the following complexity classification: Unweighted Stars NWS is FPT with respect to $c$ and Stars NWS is W[1]-hard in the most restricted case when $G$ is a clique and all edges have weight 1 or 2 and in XP in the most general case.

For an overview of the parameterized complexity results, refer to Table 1.

Due to lack of space several proofs (marked with ($\star$)) are deferred to the full version.

## 2 Preliminaries and Basic Observations

**Preliminaries.** For a set $X$, we denote by $\binom{X}{2}$ the collection of all size-two subsets of $X$. Moreover, for positive integers $i$ and $j$ with $i \le j$, we denote by $[i, j] := \{k \in \mathbb{N} : i \le k \le j\}$.

An *undirected graph* $G = (V, E)$ consists of a set of vertices $V$ and a set of edges $E \subseteq \binom{V}{2}$. We denote by $V(G)$ and $E(G)$ the vertex and edge set of $G$, respectively, and let $n = n(G) := |V(G)|$ and $m := |E(G)|$. For a vertex set $V' \subseteq V$, we denote by $E_G(V') := \{\{u, v\} \in E : u, v \in V'\}$ the edges between the vertices of $V'$ in $G$. If $G$ is clear from the context, we may omit the subscript. A graph $G'$ is a *subgraph* of $G$ if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. Moreover, $G'$ is *spanning* if $V(G') = V(G)$. For a vertex set $V'$, we denote by $G[V'] := (V', E_G(V'))$ the *subgraph of $G$ induced by $V'$*. A set $S \subseteq V(G)$ with

[1] The parameter $t$ can be computed in polynomial time as discussed in Section 2.

$E_G(S) = \binom{S}{2}$ is called a *clique*. A graph $G$ is a *star* of size $n-1$ with *center* $z \in V(G)$ if $E(G) = \{\{z, v\} : v \in V(G) \setminus \{z\}\}$. A graph $G$ contains a *spanning star* if some subgraph $G'$ of $G$ is a star of size $n-1$. The center of this star is *universal* for $G$.

An edge set $E' \subseteq E(G)$ is a *feedback edge set* of $G$, if the graph $G' := (V(G), E(G) \setminus E')$ is acyclic. Two vertices $u$ and $v$ are *connected* in $G$ if $G$ contains a path between $u$ and $v$. A graph $G$ is *connected* if each pair of vertices $u, v \in V(G)$ is connected. A set $S \subseteq V(G)$ is a *connected component* of $G$, if $G[S]$ is connected and $S$ is maximal with this property. Connectivity in hypergraphs is defined similarly: Two vertices $u$ and $v$ are *connected* if there exists a sequence $C_1, C_2, \ldots, C_p$ of hyperedges such that $u \in C_1$, $v \in C_p$, and consecutive communities have nonempty intersection. A *connected component* of a hypergraph is a maximal set of connected vertices. The number $x$ of connected components of a hypergraph can be computed in polynomial time, for example by BFS. Note that for a minimal solution $G'$ for Stars NWS and Connectivity NWS, the connected components of $G'$ are exactly the connected components of the community hypergraph. Thus, $t = \ell - n + x$ and the parameter $t$ can be computed in polynomial time for a given input instance.

For details about parameterized complexity and the ETH refer to [12, 11].

**Basic observations.** To put our main results for Connectivity NWS and Stars NWS into context, we state some results that either follow easily from previous work or from simple observations.

The naive brute-force approach for each $\Pi$-NWS is to perform an exhaustive search over the $\mathcal{O}(2^m)$ possibilities to select at most $\ell$ edges from the input graph $G$. This leads to the following general statement for $\Pi$-NWS problems.

▶ **Proposition 2.1.** *Let $\Pi$ be a property which can be decided in $\text{poly}(n)$ time. Then, $\Pi$-NWS is solvable in $2^m \cdot c \cdot \text{poly}(n)$ time.*

For the solution size parameter $\ell$, one can obtain the following running time.

▶ **Proposition 2.2** ($\star$). *Connectivity NWS and Stars NWS can be solved in $\ell^{\mathcal{O}(\ell)} \cdot \text{poly}(n+c)$ time.*

The fixed-parameter tractability of Unweighted Connectivity NWS with respect to $\ell$ was also shown by Fluschnik and Kellerhals via a kernelization [17].

A further natural parameter that can be considered is $k := m - \ell$, a lower-bound on the number of edges of $G$ that any solution must omit.

▶ **Proposition 2.3** ($\star$).
- *Connectivity NWS and Stars NWS are NP-hard for $k = 0$.*
- *Unweighted Connectivity NWS and Unweighted Stars NWS can be solved in $n^{2k} \cdot \text{poly}(n)$ time and are $\text{W}[1]$-hard with respect to $k$ even if $G$ is a clique and if each community has size at most 3.*

For Unweighted Connectivity NWS, Fluschnik and Kellerhals showed that a polynomial kernel for $\ell$ and (thus for $n$) is unlikely, even on planar series-parallel graphs [17]. This result can be also given for the case when the input graph $G$ is a clique.

▶ **Proposition 2.4** ($\star$). *Unweighted Connectivity NWS and Unweighted Stars NWS do not admit a polynomial kernel for $n$ even if $G$ is a clique, unless NP $\subseteq$ coNP/poly.*

One can also show that the simple brute-force algorithm that considers all $\mathcal{O}(2^m)$ subsets of $E(G)$ cannot be improved substantially.

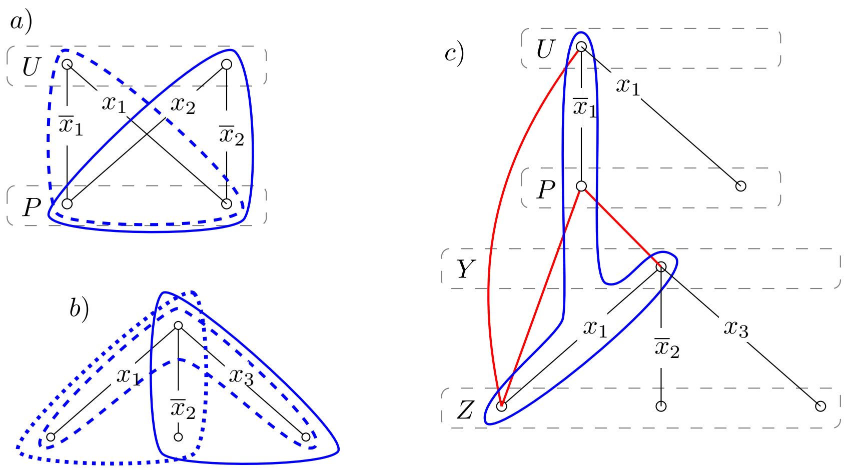


**Figure 2** Sketch of the construction of Theorem 3.1. The communities are blue (solid, dashed and dotted). We only show edges which are contained in at least one community and only some fixed edges (red). *a)* Part of the variable gadget for $x_1$ and $x_2$. *b)* The variable communities for a clause $q = x_1 \vee \overline{x_2} \vee x_3$. *c)* The assignment gadget for the first literal $x_1$ of the clause $q$. Here, the red edges are the fixed edges with one endpoint in the variable gadget and one in the clause gadget.

▶ **Proposition 2.5** ($\star$). *If the ETH is true, then* Unweighted Stars NWS *and* Unweighted Connectivity NWS *cannot be solved in* $2^{o(n+m+c)} \cdot \mathrm{poly}(n+c)$ *time, even if restricted to instances with community size at most* 3.

## 3 A Stronger ETH-Bound

In Proposition 2.5 we observed that algorithms with running time $2^{o(n+m+c)}$ for Unweighted Connectivity NWS and Unweighted Stars NWS would violate the ETH. We now provide a stronger $2^{\Omega(n^2+c)}$-time lower bound for both problems. Notably, this lower bound also applies to the case when all communities have constant size.

First, we present the lower bound for Unweighted Stars NWS.

▶ **Theorem 3.1.** *If the ETH is true, then* Unweighted Stars NWS *cannot be solved in* $2^{o(n^2+c)}$ *time, even if $G$ is a clique and if each community has size at most* 4.

**Proof.** We reduce from 3-SAT to Unweighted Stars NWS such that the resulting instance has $\mathcal{O}(\sqrt{|\phi|})$ vertices and $\mathcal{O}(|\phi|)$ communities, where $\phi$ denotes the total formula length. Then, the existence of an $2^{o(n^2+c)}$-time algorithm for Unweighted Stars NWS implies the existence of a $2^{o(|\phi|)}$-time algorithm for 3-SAT violating the ETH [25, 26]. The input formula $\phi$ is over the variable set $X$ and each clause $q \in \Gamma$ contains exactly three literals. For a literal $y$, we denote by $\overline{y}$ its complement. A visualization of the construction is given in Figure 2. In all gadgets, we add several communities of size 2. These communities enforce that each solution has to contain the edge of this community. In the following we call such edges *fixed*.

**Variable gadget $G_X$.** Recall that $G_X$ is a clique. The idea is to create for each variable a community $C$ of size 3 with one *fixed* edge. The two remaining edges of $C$ are called *selection* edges. The idea is that each solution contains exactly one selection edge of $C$. One selection edge represents the positive literal, the other one represents the negative literal. The fixed edge of the triangle is used to model that one literal must be set to `true`. The selection edges are arranged compactly, to guarantee that $|V(G_X)| \in \mathcal{O}(\sqrt{|\phi|})$. In the following, we describe the graph $G_X$ together with communities fulfilling the above-described properties. An example of a variable gadget is shown in part *a)* of Figure 2.

Let $V(G_X) = U \cup P$ where $U := \{u_1, \ldots, u_{n_x}\}$, $P = P_1 \cup P_2$, and $P_i := \{p_1^i, \ldots, p_{n_x}^i\}$ for $i \in [2]$ consist of $n_x = \lceil \sqrt{|X|} \rceil$ vertices each. It remains to describe the communities: For each variable $x \in X$, we add a community $C_x := \{u_j, p_s^1, p_s^2\}$ for $j, s \in [\, n_x \,]$. This is possible since $n_x \cdot n_x \geq |X|$. These communities are called the *variable communities* $\mathcal{C}^X$. Afterwards, we set $\theta(x) := \{u_j, p_s^1\}$ and $\theta(\overline{x}) := \{u_j, p_s^2\}$ to assign the positive and negative literal of $x$ to an edge of the variable gadget. Now, we fix the edges of $G[P] = G[P_1 \cup P_2]$. In other words, for each edge $\{p_{j_1}^{i_1}, p_{j_2}^{i_2}\}$ having both endpoints in $P_1 \cup P_2$, we add a community $\{p_{j_1}^{i_1}, p_{j_2}^{i_2}\}$.

Observe that the sets of selection edges corresponding to two distinct variables are disjoint:

▷ Claim 1 ($\star$). Each selection edge of $E(G_X)$ is contained in only one subgraph induced by a variable community in $\mathcal{C}^X$.

**Clause gadget.** We continue by describing the construction of the clause gadget $G_\Gamma$. The idea is that each clause is represented by four vertices of $V(G_\Gamma)$ in which a triangle is *fixed*. All three remaining edges of this size-4 clique are referred to as *free*. Note that these free edges form a star with three leaves. Each free edge represents one literal of the clause. For each pair containing two of these three edges, we then create a community containing the three endpoints of these two edges. As in the vertex gadget, these induced subgraphs are arranged compactly, to achieve a clause gadget with $|V(G_\Gamma)| \in \mathcal{O}(\sqrt{|\Gamma|})$.

Let $V(G_\Gamma) = Y \cup Z$ where $Y = \{y_1, \ldots, y_{n_c}\}$, $Z = Z_1 \cup Z_2 \cup Z_3$, and $Z_i = \{z_1^1, \ldots, z_{n_c}^i\}$ for $i \in [3]$ consist of $n_c = \lceil \sqrt{|\Gamma|} \rceil$ vertices each. In the following, we assign each clause to a clique of $G_\Gamma$ having vertex set $y_j, z_s^1, z_s^2, z_s^3$ for $j, s \in [n_c]$. This is possible since $n_c \cdot n_c \geq |\Gamma|$. In this clique, we *fix* the triangle having its endpoints in $Z_1 \cup Z_2 \cup Z_3$. Formally, for each clause $q = \{q_1, q_2, q_3\} \in \Gamma$ we add three communities $C_q^1 = \{y_j, z_s^2, z_s^3\}$, $C_q^2 = \{y_j, z_s^1, z_s^3\}$ and $C_q^3 = \{y_j, z_s^1, z_s^2\}$. We refer to these communities as the *clause* communities $\mathcal{C}^\Gamma$. Afterwards, we set $\nu(q, q_1) := \{y_j, z_s^1\}$, $\nu(q, q_2) := \{y_j, z_s^2\}$, and $\nu(q, q_3) := \{y_j, z_s^3\}$ to assign each literal in clause $q$ to an edge of the clause gadget. These edges are referred to as *free*. Second, we fix the edges of the clique $Z_1 \cup Z_2 \cup Z_3$.

Observe that the sets of free edges corresponding to two distinct clauses are disjoint:

▷ Claim 2 ($\star$). Each free edge of $E(G_\Gamma)$ is contained in exactly one subgraph induced by a clause community in $\mathcal{C}^\Gamma$.

**Connecting the gadgets.** We complete the construction by connecting the variable and clause gadget, using new *assignment* communities. The idea is to add a new community containing the endpoints of a free edge describing a literal in a clause together with the endpoints of the selection edge describing the same opposite literal in the variable gadget. These communities model occurrences of variables in the clauses. Roughly speaking, these communities are satisfied if the selection edge of the variable gadget or the free edge of the clause gadget is part of the solution. To enforce this, we fix further edges of $G$.

We create for each clause $q = \{q_1, q_2, q_3\} \in \Gamma$ three assignment communities $C_q^{q_1} = \nu(q, q_1) \cup \theta(\overline{q_1})$, $C_q^{q_2} = \nu(q, q_2) \cup \theta(\overline{q_2})$, and $C_q^{q_3} = \nu(q, q_3) \cup \theta(\overline{q_3})$. We denote the assignment communities with $\mathcal{C}_\Gamma^X$. To enforce that each solution contains the selection edge or the free edge of each assignment community, we fix all edges between the vertex sets $U$ and $Z$, between the vertex sets $P$ and $Y$, and between the vertex sets $P$ and $Z$.

Finally, we set $\ell := |X| + 2 \cdot |\Gamma| + \binom{|P|}{2} + \binom{|Z|}{2} + |U| \cdot |Z| + |P| \cdot |Y| + |P| \cdot |Z|$.

**Correctness.** The correctness is based on the facts that each solution for $I$ contains *a*) all fixed edges, *b*) exactly $|X|$ selection edges, and *c*) exactly $2|\Gamma|$ free edges. Fact *b*) ensures that this models an assignment $\beta$ of the variables of $X$ and fact *c*) ensures that each clause is satisfied by at least one literal of $\beta$. The detailed correctness proof is deferred to the full version. ◀

An adapted construction yields further results for $d$-Diam NWS [17] ($\Pi$ is "having diameter at most $d$") and Density NWS [19] ($\Pi$ is "exceeding some density threshold").

▶ **Corollary 3.2** ($\star$). *If the ETH is true, then Unweighted Connectivity NWS, $d$-Diam NWS for each $d \geq 2$, and Density NWS cannot be solved in $2^{o(n^2+c)}$ time even if $G$ is a clique and each community has size at most 4.*

▶ **Corollary 3.3** ($\star$). *Unweighted Stars NWS remains NP-hard and, assuming the ETH, cannot be solved in $2^{o(n+m+c)}$ time on graphs with maximum degree six and community size at most 4.*

# 4 Parameterization by the Feedback Edge Number of a Solution

The parameter $\ell$, the number of edges in the solution is in most cases not independent from the size of the input instance of Stars NWS or Connectivity NWS: if the hypergraph $(V, \mathcal{C})$ is connected, a solution $G'$ has at least $n-1$ edges. In other words, $n-1$ is a lower bound for $\ell$ in this case. In this section, we study Stars NWS and Connectivity NWS parameterized above this lower bound. Formally, the parameter $t$ is defined as the size of a minimum feedback edge set of any optimal solution of an instance of Stars NWS or Connectivity NWS. Thus, the parameter $t$ measures how close the solution is to a forest. Formally, the definition is $t \coloneqq \ell - n + x$ where $x$ denotes the number of connected components of $G'$. Recall that $t$ can be computed in polynomial time (see Section 2.).

## 4.1 An XP-Algorithm for Stars NWS

In this subsection, we show that Stars NWS parameterized by $t$ admits an XP-algorithm.

▶ **Theorem 4.1.** *Stars NWS can be solved in $m^{4t} \cdot \mathrm{poly}(|I|)$ time.*

Our XP-algorithm exploits the fact that there are two different kinds of cycles in $G'$: First, there are *global* cycles. These are the cycles in the solutions that are directly caused by cycles in the input hypergraph. No solution may avoid these cycles. Second, there are *local* cycles. These are cycles which are entirely contained in the subgraph induced by two communities. Since in each solution, each community contains a spanning star, local cycles can only have length 3 or 4. This allows us to bound the number of possible local cycles and thus to consider all possibilities for the local cycles in XP-time with respect to $t$. Then, the crux of our algorithm is that after all local cycles have been fixed, all remaining cycles added by our algorithm have to be global and are thus unavoidable. Using this fact, we show that in polynomial time we can compute an optimal solution with feedback edge number at most $t$ that extends a fixed set of local cycles without introducing any further local cycles. To do this, for each community $C$, we store a set of *potential centers*, that is, vertices of $C$ that may be the center of a spanning star of $C$ in any solution that does not produce new local cycles. We define several *operations* that restrict the potential centers of each community. We show that after all operations have been applied exhaustively, one can greedily pick the best remaining center for each community.

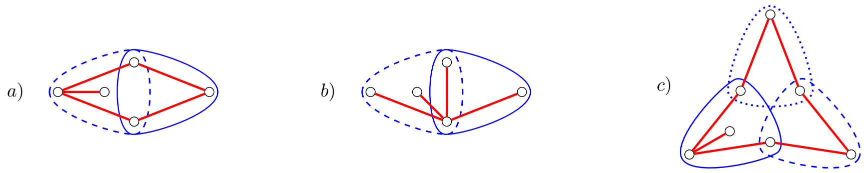


**Figure 3** Examples for solutions with and without local cycles. Red edges indicate the edges of the solution. Part a) shows an example, where both communities induce a local cycle. Part b) shows an example, where the two communities do not induce local cycle. Finally, part c) shows an example, where the solution contains a (global) cycle but no two communities induce a local cycle.

**Algorithm-specific notation.** Next, we present the formal definition of local cycles; an example is shown in Figure 3. For a spanning subgraph $H$ of $G$ and a community $C \in \mathcal{C}$, let $\mathrm{univ}_H(C)$ denote the vertices of $C$ that are *universal* for $C$ in $H$. Note that $\mathrm{univ}_H(C) \subseteq \mathrm{univ}_G(C)$. In the following, we assume that for each community $C \in \mathcal{C}$, $\mathrm{univ}_G(C) \neq \emptyset$, as otherwise, there is no solution for the instance $I$ of Stars NWS.

For a solution $G'$, we say that two distinct communities $C_1$ and $C_2$ *induce a local cycle* if for each $i \in \{1, 2\}$, there is a vertex $c_i \in \mathrm{univ}_{G'}(C_i)$ such that the graph $S_1 \cup S_2$ contains a cycle. Here, for each $i \in \{1, 2\}$, $S_i$ is the spanning star of $C_i$ with center $c_i$ and $S_1 \cup S_2$ is the union of both these stars defined by $S_1 \cup S_2 := (C_1 \cup C_2, \{\{c_i, w_i\} : w_i \in C_i \setminus \{c_i\}, i \in \{1, 2\}\})$. Moreover, we say that each cycle of $S_1 \cup S_2$ is a *local cycle* in $G'$. Note that each local cycle has length at most four, and if $C_1$ and $C_2$ induce a local cycle, then $|C_1 \cap C_2| \geq 2$.

As described above, the first step of the algorithm behind Theorem 4.1 is to test each possibility for the local cycles of the solution. For a fixed guess, we let $E^*$ denote the set of all edges contained in at least one local cycle and in the following we refer to them as *local edges*. Moreover, we call a minimum solution $G'$ *fitting for* $E^*$ if each local cycle of $G'$ uses only edges of $E^*$ and each edge of $E^*$ is contained in $G'$. Hence, to determine whether the choice of local edges $E^*$ can lead to a solution, we only have to check, whether there is a fitting solution for $E^*$. In the following, we show that this can be done in polynomial time.

▶ **Theorem 4.2.** *Let $I = (G = (V, E), \mathcal{C}, \omega, \ell, b)$ be an instance of Stars NWS, and let $E^* \subseteq E$. In polynomial time, we can*
- *find a solution $G' = (V, E')$ for $I$ with $E^* \subseteq E'$ or*
- *correctly output that there is no minimum solution that is fitting for $E^*$.*

Based on the definition of fitting solutions, we define for each community $C \in \mathcal{C}$ a set $\mathrm{fit}_{E^*}(C)$ of possible centers. We initialize $\mathrm{fit}_{E^*}(C) := \mathrm{univ}_G(C)$ for each community $C \in \mathcal{C}$. The goal is to reduce $\mathrm{fit}_{E^*}(C)$ of each community $C$ as much as possible while preserving the following property, which trivially holds for the initial $\mathrm{fit}_{E^*}(C)$ for each community $C \in \mathcal{C}$.

▶ **Property 1.** *For each minimum solution $G'$ which is fitting for $E^*$ and each community $C \in \mathcal{C}$, we have $\mathrm{univ}_{G'}(C) \subseteq \mathrm{fit}_{E^*}(C)$.*

Note that if Property 1 is fulfilled and if $\mathrm{fit}_{E^*}(C) = \emptyset$ for some $C \in \mathcal{C}$, then we can correctly output that there is no fitting solution for $E^*$. Next, we define several operations that for some communities $C \in \mathcal{C}$ remove vertices from $\mathrm{fit}_{E^*}(C)$ which – when taken as a center vertex for $C$ – would introduce new local cycles, violating the properties of a fitting solution. We show that these operations preserve Property 1 and that after these operations are applied exhaustively, the task of Theorem 4.2 can be performed greedily based on $\mathrm{fit}_{E^*}$. Examples for each of our operations are shown in Figure 5.

In the following, we say that a vertex $v \in V$ is *locally universal* for a vertex set $A \subseteq V$, if for each vertex $w \in A \setminus \{v\}$, the vertex pair $\{v, w\}$ is a local edge.

▶ **Operation 1.** *Let $C \in \mathcal{C}$ be a community and let $\{y, z\} \subseteq C$ be a local edge. Remove each vertex $v$ from $\operatorname{fit}_{E^*}(C)$ which is not locally universal for $\{y, z\}$.*

The following lemma shows that Operation 1 preserves Property 1.

▶ **Lemma 4.3** (⋆). *Let $G'$ be a minimum solution for $I$, let $C$ be a community of $\mathcal{C}$ and let $x \in \operatorname{univ}_{G'}(C)$ such that $x$ is not locally universal for some local edge $\{y, z\} \subseteq C$. Then, $G'$ is not fitting for $E^*$.*

Note that after the exhaustive application of Operation 1, for each community $C \in \mathcal{C}$ with at least one local edge, the vertices of $\operatorname{fit}_{E^*}(C)$ induce a clique with only local edges.

Next, we define a partition $\mathfrak{C}$ of the communities of $\mathcal{C}$. The idea of this partition is that in each fitting solution for $E^*$, all communities of the same part of the partition $\mathfrak{C}$ have the same unique center. The definition of the partition $\mathfrak{C}$ is based on the following lemma.

▶ **Lemma 4.4** (⋆). *Let $C$ and $D$ be distinct communities of $\mathcal{C}$ with $|C \cap D| \geq 3$ and where no vertex $v \in C \cup D$ is locally universal for $C \cap D$. Let $G'$ be a solution such that there is no vertex $w \in C \cap D$ with $\operatorname{univ}_{G'}(C) = \operatorname{univ}_{G'}(D) = \{w\}$. Then, $C$ and $D$ induce a local cycle in $G'$ that uses at least one edge which is not a local edge.*

Consider the auxiliary graph $G_{\mathfrak{C}}$ with vertex set $\mathcal{C}$ and where two distinct communities $C$ and $D$ are adjacent if and only if *a*) $|C \cap D| \geq 3$ and *b*) there is no locally universal vertex for $C \cap D$ in $C \cup D$. The partition $\mathfrak{C}$ consists of the connected components of $G_{\mathfrak{C}}$ and for a community $C \in \mathcal{C}$, we denote by $\mathfrak{C}(C)$ the collection of communities in the connected component of $C$ in $G_{\mathfrak{C}}$. An example is shown in Figure 4.

By Lemma 4.4 and due to transitivity, we obtain the following.

▶ **Corollary 4.5.** *For each community $C \in \mathcal{C}$ with $|\mathfrak{C}(C)| \geq 2$ and each fitting solution $G'$ for $E^*$, there is a vertex $v \in \bigcap_{\widetilde{C} \in \mathfrak{C}(C)} \widetilde{C}$ such that $\operatorname{univ}_{G'}(\widetilde{C}) = \{v\}$ for each $\widetilde{C} \in \mathfrak{C}(C)$.*

This implies that the following operation preserves Property 1.

▶ **Operation 2.** *Let $C \in \mathcal{C}$. Remove each vertex $v$ from $\operatorname{fit}_{E^*}(C)$ if $v$ is not contained in $\bigcap_{\widehat{C} \in \mathfrak{C}(C)} \operatorname{fit}_{E^*}(\widehat{C})$.*

Next, we define an operation for each possibility how two communities may intersect.

▶ **Operation 3.** *Let $C \in \mathcal{C}$ such that $C$ contains no local edge. Moreover, let $D \in \mathcal{C}$ such that $|C \cap D| \geq 2$. Remove all vertices from $\operatorname{fit}_{E^*}(C)$ that are not contained in $C \cap D$.*

▶ **Operation 4.** *Let $C \in \mathcal{C}$ such that $C$ contains at least one local edge. Moreover, let $D \notin \mathfrak{C}(C)$ be a community, such that $|C \cap D| = 2$ and $\{x, y\} := C \cap D$ is not a local edge.*
1. *If $\operatorname{fit}_{E^*}(C) \cap \{x, y\} = \emptyset$, then remove $x$ and $y$ from $\operatorname{fit}_{E^*}(D)$ or*
2. *if $\operatorname{fit}_{E^*}(C) \cap \{x, y\} = \{x\}$, then set $\operatorname{fit}_{E^*}(C) := \{x\}$.*

▶ **Operation 5.** *Let $C \in \mathcal{C}$ be a community containing at least one local edge. Moreover, let $D \notin \mathfrak{C}(C)$ such that $|C \cap D| \geq 3$. For each pair of distinct vertices $x$ and $y$ of $C \cap D$, where $\{x, y\}$ is not a local edge, remove $x$ and $y$ from $\operatorname{fit}_{E^*}(D)$.*

▶ **Lemma 4.6** (⋆). *Operation 3 preserves Property 1. Moreover, if Operation 1 is exhaustively applied, then Operation 4 and Operation 5 preserve Property 1.*

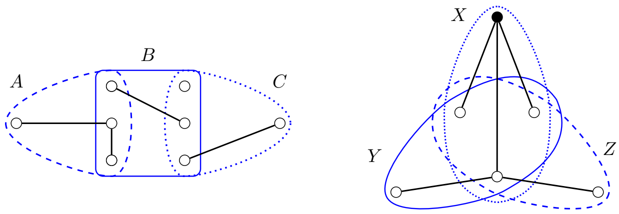


**Figure 4** Examples for parts of the partition $\mathfrak{C}$. Only the local edges are shown. Note that $A, C \in \mathfrak{C}(B)$, since $A$ and $C$ share at least three vertices with $B$ and no vertex of $A \cup B$ or $C \cup B$ is locally universal for $A \cap B$ or $C \cap B$, respectively. Hence, after exhaustive application of Operation 2, $\text{fit}_{E^*}(A) = \text{fit}_{E^*}(B) = \text{fit}_{E^*}(C) = \emptyset$, since $A$ and $C$ share no vertices. Furthermore, $Y \in \mathfrak{C}(Z)$, since no vertex of $Y \cup Z$ is locally universal for $Y \cap Z$. Note that $X \notin \mathfrak{C}(Z)$, since the black vertex of $X$ is locally universal for $X \cap Y$ and $X \cap Z$. Observe that an exhaustive application of Operation 2 yields $\text{fit}_{E^*}(Z) \subseteq Y \cap Z$ and an exhaustive application of Operation 5 yields $\text{fit}_{E^*}(Z) \cap (X \cap Z) = \text{fit}_{E^*}(Z) \cap (Y \cap Z) = \emptyset$, since $X$ contains at least one local edge and $X \cap Z$ contains no local edge. Hence, for both shown hypergraphs, there is no fitting solution for the given set of local edges.

**Algorithm 1** Algorithm solving the problem described in Theorem 4.2.

**Input** : $I = (G = (V, E), \mathcal{C}, \omega, \ell, b), E^* \subseteq E$
**Output**: A solution $G' = (V, E')$ with at most $\ell$ edges and total weight at most $b$, or no, if there is no minimal solution which is fitting for $E^*$

Compute the partition $\mathfrak{C}$ of $\mathcal{C}$
For each $C \in \mathcal{C}$, initialize $\text{fit}_{E^*}(C) \leftarrow \text{univ}_G(C)$ and apply Operation 1
Apply Operations 1–5 exhaustively
**if** $\text{fit}_{E^*}(C) = \emptyset$ *for some* $C \in \mathcal{C}$ **then return** *no*
$G_A \leftarrow (V, E^*)$
**forall** $\mathcal{L} \in \mathfrak{C}$ **do**
  $C \leftarrow$ some community of $\mathcal{L}$
  $V_{\mathcal{L}} \leftarrow \bigcup_{\widetilde{C} \in \mathcal{L}} \widetilde{C}$
  $y \leftarrow \arg\min_{u \in \text{fit}_{E^*}(C)} \omega(\{\{u, v\} : v \in V_{\mathcal{L}} \setminus \{u\}\} \setminus E^*)$
  add all edges of $\{\{y, v\} : v \in V_{\mathcal{L}} \setminus \{y\}\}$ to $G_A$
**if** $|E(G_A)| \leq \ell$ *and* $\omega(E(G_A)) \leq b$ **then return** $G_A$
**return** *no*

Based on these operations, we are now able to present the algorithm (see Algorithm 1) behind Theorem 4.2 working as follows: First, we apply Operations 1–5 exhaustively. Next, if there is a community $C \in \mathcal{C}$ with $\text{fit}_{E^*}(C) = \emptyset$, then we return no. This is correct, since Operations 1–5 preserve Property 1. Afterwards, we start with an auxiliary graph $G_A$ with vertex set $V$ and edge set $E^*$ and we iterate over the partition $\mathfrak{C}$. Recall that since Operation 2 is exhaustively applied, for each $\mathcal{L} \in \mathfrak{C}$, $\text{fit}_{E^*}(C) = \text{fit}_{E^*}(D)$ for any two communities $C$ and $D$ of $\mathcal{L}$. For each $\mathcal{L} \in \mathfrak{C}$, we find a vertex $y \in \text{fit}_{E^*}(C)$ that minimizes the total weight of non-local edges required to make $y$ the center of all communities of $\mathcal{L}$, where $C$ is an arbitrary community of $\mathcal{L}$. Finally, we add all edges between $y$ and each vertex of any community of $\mathcal{L}$ to $G_A$. After the iteration over the partition $\mathfrak{C}$ is completed, we output $G_A$ if it contains at most $\ell$ edges and has total weight at most $b$. Otherwise, we return that there is no fitting solution for $E^*$. It remains to show that this greedy choice for the center vertices is correct.

▶ **Lemma 4.7.** *Algorithm 1 is correct.*

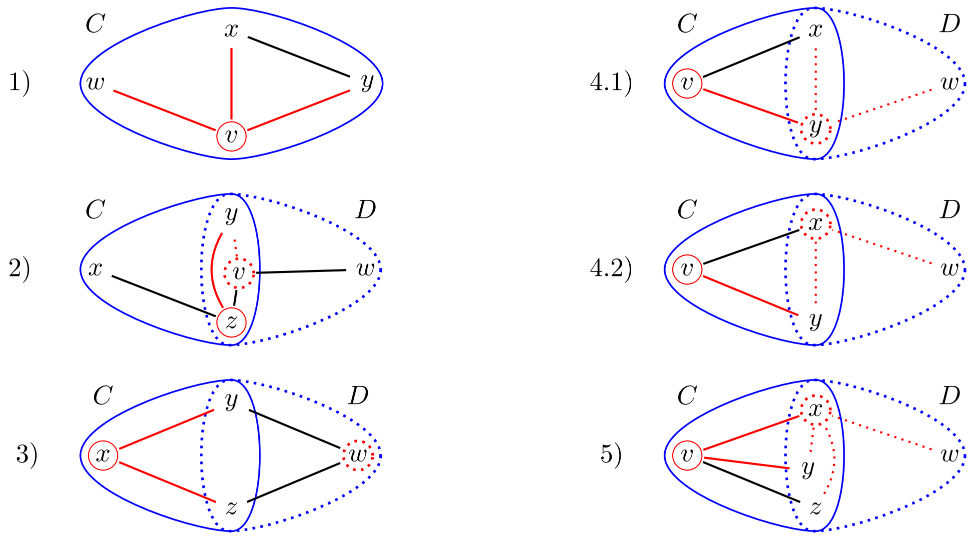


**Figure 5** Examples of applications of Operations 1–5. The black edges represent the local edges, the solid (for $C$) or dashed (for $D$) red edges show the non-local edges resulting from choosing the respective center for community $C$ or $D$. For example in 2), $z$ is the center of community $C$ and $v$ is the center of community $D$, and the edges $\{z, y\}$ and $\{v, y\}$ are non-local edges in the solution. For each operation, the violation of the property of being a fitting solution is shown, if a vertex $a$ is selected as a center of a community $A$ where the application of the corresponding operation would remove $a$ from $\text{fit}_{E^*}(A)$. In 1), 2), 3), and 4.2), the vertex selected as center for community $C$ is removed from $\text{fit}_{E^*}(C)$ by the respective operation. For example, in 4.2), (assuming $\text{fit}_{E^*}(C) \cap \{x, y\} = \{x\}$) Operation 4 removes $v$ from $\text{fit}_{E^*}(C)$, as otherwise selecting $v$ as center of $C$ results in the depicted non-fitting solution. In 4.1) and 5), the vertex selected as center for community $D$ is removed from $\text{fit}_{E^*}(D)$ by the respective operation. For example in 4.1), (assuming $\text{fit}_{E^*}(C) \cap \{x, y\} = \emptyset$) Operation 4 removes $y$ from $\text{fit}_{E^*}(D)$, as otherwise selecting $y$ as center of $D$ results in the depicted non-fitting solution.

**Proof.** If Algorithm 1 outputs "no" in Line 4, then this is correct, since $\text{fit}_{E^*}$ fulfills Property 1. Otherwise, let $G_A$ denote the graph constructed by Algorithm 1 and let for each community $C \in \mathcal{C}$, $\text{center}(C)$ denote the vertex $y$ chosen to be the center of all communities of $\mathfrak{C}(C)$ in Line 9. By construction, $G_A$ is a solution since for each community $C \in \mathcal{C}$, $\text{center}(C)$ is a vertex of $\text{fit}_{E^*}(C) \subseteq \text{univ}_G(C)$. If $G_A$ contains at most $\ell$ edges and has total weight at most $b$, then the algorithm correctly outputs the solution $G_A$ which is fitting for $E^*$.

Thus, in the following we assume that $G_A$ contains more than $\ell$ edges or has weight more than $b$. Assume towards a contradiction that there is a fitting solution $G_F$ for $E^*$ such that $\text{Agree}(G_F) := \{C \in \mathcal{C} : \text{center}(C) \in \text{univ}_{G_F}(C)\}$ is as large as possible.

**Case 1.** $\text{Agree}(G_F) = \mathcal{C}$. By construction, $G_A$ contains all edges of $E^*$ and only the required edges to achieve that for each community $C \in \mathcal{C}$, $\text{center}(C) \in \text{univ}_{G_A}(C)$. Consequently, $G_A$ is a subgraph of $G_F$ and thus $G_F$ contains more than $\ell$ edges or has weight more than $b$, a contradiction.

**Case 2.** There is a community $C \in \mathcal{C} \setminus \text{Agree}(G_F)$. In the following, we define a fitting solution $G'_F$ for $E^*$ with $\text{Agree}(G'_F) \supsetneq \text{Agree}(G_F)$. By definition, $\text{center}(C) = \text{center}(\widetilde{C})$ for each community $\widetilde{C} \in \mathfrak{C}(C)$. Let $V_C := \bigcup_{\widetilde{C} \in \mathfrak{C}(C)} \widetilde{C}$ and let $y := \text{center}(C)$. Moreover, let $x$ be an arbitrary vertex of $V_C$ such that $x \in \text{univ}_{G_F}(\widetilde{C})$ for each community $\widetilde{C} \in \mathfrak{C}(C)$. Due to Corollary 4.5 and since $G_F$ is fitting for $E^*$, this vertex exists and is unique if $|\mathfrak{C}(C)| \geq 2$. Note that $C \in \mathcal{C} \setminus \mathrm{Agree}(G_F)$ implies that $x \neq y$. This also implies that $C$ has size at least 3, and thus, each community of $\mathfrak{C}(C)$ has size at least 3. We obtain $G'_F$ as follows: First, initialize $G'_F$ as $G_F$. Second, for each community $\widetilde{C} \in \mathfrak{C}(C)$, remove all edges that are not local edges of $G_F[\widetilde{C}]$ from $G'_F$. Finally, for each community $\widetilde{C} \in \mathfrak{C}(C)$, add the minimum number of edges to $G'_F$ such that $y \in \mathrm{univ}_{G'_F}(\widetilde{C})$, that is, the edges $\{\{y, v\} : v \in V_C \setminus \{y\}\} \setminus E^*$.

First, we show that $G'_F$ contains at most as many edges as $G_F$. To this end, we first observe the following.

▷ Claim 3 (⋆). For each $z \in V_C \setminus \{x, y\}$, the edge $\{x, z\}$ is a local edge if and only if $\{y, z\}$ is a local edge.

Recall that each edge which is in $G'_F$ and not in $G_F$ is incident with $y$ and some vertex of $V_C \setminus \{x, y\}$. Hence, for each $z \in V_C \setminus \{x, y\}$ where the edge $\{y, z\}$ was added to obtain $G'_F$, the edge $\{x, z\}$ was removed to obtain $G'_F$. Thus, $G'_F$ contains at most as many edges as $G_F$. This implies that the difference between the total weight of $G'_F$ and the total weight of $G_F$ is at most $\rho = \omega(\{\{y, z\} : z \in V_C \setminus \{y\}\} \setminus E^*) - \omega(\{\{x, z\} : z \in V_C \setminus \{x\}\} \setminus E^*)$. Due to Line 9, $\rho$ is not positive. Thus, since $G_F$ has total weight at most $b$, $G'_F$ has total weight at most $b$.

To show that $G'_F$ is a solution, it remains to show that each community $C \in \mathcal{C}$ has at least one center in $G'_F$. For this, it suffices to show that all communities outside of $\mathfrak{C}(C)$ have the same centers in $G_F$ and $G'_F$, since $y$ is a center of all communities of $\mathfrak{C}(C)$.

▷ Claim 4. For each community $D \in \mathcal{C} \setminus \mathfrak{C}(C)$, $\mathrm{univ}_{G_F}(D) = \mathrm{univ}_{G'_F}(D)$.

Proof. Due to symmetry, we only show that $\mathrm{univ}_{G_F}(D) \subseteq \mathrm{univ}_{G'_F}(D)$. Assume towards a contradiction that there is a vertex $z \in \mathrm{univ}_{G_F}(D) \setminus \mathrm{univ}_{G'_F}(D)$. Since $z \notin \mathrm{univ}_{G'_F}(D)$, there is an edge $\{z, w\}$ which is contained in $G_F$ but not in $G'_F$. Moreover, $\{z, w\}$ is not a local edge, since $G'_F$ contains all local edges. This further implies that there is a community $\widetilde{C} \in \mathfrak{C}(C)$ such that $\{z, w\} \subseteq \widetilde{C}$. Since $G_F$ is fitting for $E^*$, $x$ is one endpoint of $\{z, w\}$, as otherwise, $\widetilde{C}$ and $D$ induce a local cycle in $G_F$ on the vertices $\{x, z, w\}$ and the edge $\{z, w\}$ is not a local edge. Next, we distinguish the cases whether $\widetilde{C}$ contains a local edge.

**Case 1.** There is no local edge in $\widetilde{C}$. Since Operation 3 is exhaustively applied, $\{x, y\} \subseteq \mathrm{fit}_{E^*}(\widetilde{C}) \subseteq \widetilde{C} \cap D$. Hence, if $|\widetilde{C} \cap D| = 2$, then $x = z$ and $y = w$, or vice versa. Consequently, the edge $\{z, w\}$ is contained in $G'_F$, a contradiction. Otherwise, assume $|\widetilde{C} \cap D| \geq 3$. We show that in this case, there is no fitting solution for $E^*$. Since $D$ is not in $\mathfrak{C}(C)$, there is some vertex of $\widetilde{C} \cup D$ which is locally universal for $\widetilde{C} \cap D$. Hence, $\mathrm{fit}_{E^*}(\widetilde{C}) \subseteq \widetilde{C} \cap D$, since $\widetilde{C}$ contains no local edge and Operation 3 is exhaustively applied. Moreover, since Operation 5 is exhaustively applied and there is no local edge between any two vertices of $\widetilde{C} \cap D$, $\mathrm{fit}_{E^*}(\widetilde{C}) \cap (\widetilde{C} \cap D) = \emptyset$. We conclude that $\mathrm{fit}_{E^*}(\widetilde{C}) = \emptyset$, which implies that there is no fitting solution for $E^*$, a contradiction to the fact that $G_F$ is a fitting solution for $E^*$.

**Case 2.** There is some local edge in $\widetilde{C}$. Recall that Operation 4 and Operation 5 are applied exhaustively with respect to $\widetilde{C}$. If $x = z$ and $y = w$, or vice versa, then the edge $\{z, w\}$ is contained in $G'_F$, a contradiction. Otherwise, let $w^*$ be the unique vertex of $\{z, w\} \setminus \{x\}$. Since $\{x, w^*\} = \{z, w\}$ is not a local edge, $x \in \mathrm{fit}_{E^*}(\widetilde{C})$, and Operation 1 is exhaustively applied, no vertex of $\mathrm{fit}_{E^*}(\widetilde{C})$ is locally universal for $w^*$ and $w^* \notin \mathrm{fit}_{E^*}(\widetilde{C})$. Hence, if $|\widetilde{C} \cap D| = 2$, then since Operation 4 is exhaustively applied, $\mathrm{fit}_{E^*}(\widetilde{C})$ has size at most one, a contradiction. Otherwise, if $|\widetilde{C} \cap D| \geq 3$, then since Operation 5 is exhaustively applied $x \notin \mathrm{fit}_{E^*}(D)$ and $w^* \notin \mathrm{fit}_{E^*}(D)$. Consequently, $z \notin \mathrm{fit}_{E^*}(D)$, a contradiction. ◁

Since $G_F$ is a solution, for each community $D \in \mathcal{C} \setminus \mathfrak{C}(C)$, Claim 4 implies that $\mathrm{univ}_{G'_F}(D) = \mathrm{univ}_{G_F}(D)$ is nonempty. Hence, $G'_F$ is a solution. Moreover, since $G_F$ is a minimum solution, Claim 3 implies that $G'_F$ is a minimum solution.

Next, we show that $G'_F$ is fitting for $E^*$. To show that $G'_F$ is a fitting solution for $E^*$, it remains to show that each local cycle of $G'_F$ uses only edges of $E^*$.

▷ Claim 5 (⋆). Each local cycle of $G'_F$ uses only edges of $E^*$.

Finally, we show that $\mathrm{Agree}(G'_F)$ is a proper superset of $\mathrm{Agree}(G_F)$. By construction, $\mathfrak{C}(C) \subseteq \mathrm{Agree}(G'_F)$, and due to Claim 4, for each community $D \in \mathcal{C} \setminus \mathfrak{C}(C)$, $\mathrm{univ}_{G'_F}(D) = \mathrm{univ}_{G_F}(D)$. Hence, $\mathrm{Agree}(G_F) \subseteq \mathrm{Agree}(G'_F)$. Moreover, since $C \notin \mathrm{Agree}(G'_F) \setminus \mathrm{Agree}(G_F)$ we obtain that $\mathrm{Agree}(G'_F)$ is a proper superset of $\mathrm{Agree}(G_F)$. Altogether, $G'_F$ is a fitting solution for $E^*$ with $\mathrm{Agree}(G'_F) \supsetneq \mathrm{Agree}(G_F)$. This contradicts our choice of $G_F$.

Hence, if $G_A$ contains more than $\ell$ edges or has weight more than $b$, then the algorithm correctly outputs that there is no solution which is fitting for $E^*$. ◀

**Proof of Theorem 4.2.** Clearly, the partition $\mathfrak{C}$ of $\mathcal{C}$ and also the initialization of $\mathrm{fit}_{E^*}$ in Lines 1 and 2 can be computed in polynomial time. Note that Operations 1–5 can be exhaustively applied in polynomial time by iterating over all local edges and all pairs of communities, since for each community $C \in \mathcal{C}$, $\mathrm{fit}_{E^*}(C)$ initially has size at most $|C| < n$ and each application of any operation may only remove elements from $\mathrm{fit}_{E^*}(C)$. Hence, Lines 3–5 can be performed in polynomial time. Afterwards, Lines 6–10 can be performed in polynomial time since for each partite set of $\mathfrak{C}$ we compute the vertex $y$ with minimal cost such that $y$ serves as the center of all communities in this partite set. Finally, the check whether the solution has at most $\ell$ edges and weight at most $b$ can be done in polynomial time. Thus, Algorithm 1 runs in polynomial time. ◀

**Finding the correct edge set $E^*$.** To solve Stars NWS, the main algorithmic difficulty now lies in finding an edge set $E^*$ that contains all edges of local cycles of any optimal solution of $I$. Hence, to prove Theorem 4.1, it remains to show that such an edge set can be found in $m^{4t} \cdot \mathrm{poly}(n + c)$ time, if it exists.

▶ **Lemma 4.8** (⋆). *If $I$ is a yes-instance of Stars NWS, then for every optimal solution $G' = (V, E')$, there is an edge set $E^* \subseteq E'$ of size at most $4t$ such that the edge set of each local cycle of $G'$ is a subset of $E^*$.*

**Proof of Theorem 4.1.** The algorithm works as follows: iterate over all possible edge sets $E^*$ of size at most $4t$ and apply the algorithm behind Theorem 4.2. If $I$ is a yes-instance, then for some edge set $E^*$, Theorem 4.2 yields an optimal solution for $I$ with at most $\ell$ edges and weight at most $b$. Since there are $\mathcal{O}(m^{4t})$ edges sets of size at most $4t$, the algorithm achieves the stated running time. ◀

**The concrete algorithm for $t = 0$.** Recall that Theorem 4.1 affirmatively answers the question by Korach and Stern [30] who asked whether there is a polynomial-time algorithm for finding an optimal solution for Stars NWS with $t = 0$. For this case, the concrete algorithm is much simpler since most of the described operations are never applicable. This is due to the fact that for $t = 0$, no local edges exist and Operations 1, 4, and 5 require at least one local edge to be applicable. In the following, we give an intuitive description of the concrete much simpler algorithm for $t = 0$.

First, we set $E^* = \emptyset$ and initialize $\mathrm{fit}_{E^*}(C) := \mathrm{univ}_G(C)$ for each community $C \in \mathcal{C}$. Afterwards, we again compute the partition $\mathfrak{C}$ of the communities of $\mathcal{C}$. Recall that this is done by defining an auxiliary graph $G_{\mathfrak{C}}$ with vertex set $\mathcal{C}$ where two distinct communities $C$ and $D$ are adjacent if and only $C$ and $D$ have an intersection of size at least 3. Note that in the original definition one had to also check that no vertex is locally optimal for the intersection. This now always holds since there are no local edges. The partition $\mathfrak{C}$ then consists of the connected components of $G_{\mathfrak{C}}$.

Second, we exhaustively apply Operations 2 and 3. Operation 2 ensures that all communities of the same part of $\mathfrak{C}$ will have the same potential centers according to $\text{fit}_{E^*}$. Moreover, Operation 3 ensures that if two communities $C$ and $D$ have an intersection of size at least 2, then the potential centers of both communities will be within $C \cap D$ according to $\text{fit}_{E^*}$.

After exhaustive application of these two operations, the remaining lines of Algorithm 1 are executed, that is, if $\text{fit}_{E^*}(C) = \emptyset$ for at least one community $C \in \mathcal{C}$, we correctly output that the instance under consideration is a no-instance of Stars NWS. Otherwise, we greedily select for each part $\mathcal{L}$ of the partition $\mathfrak{C}$ a vertex $y$ as center for all communities $\mathcal{L}$, such that $y$ is a potential center of each community of $\mathcal{L}$ and such that the cost of selecting $y$ as center of all these communities is minimum under all such potential centers. Finally, if these choices of center vertices result in more than $\ell$ edges or total weight more than $b$, we correctly output that the input instance is a no-instance of Stars NWS. Otherwise, the chosen center vertices induce a solution with $t = 0$.

## 4.2 Connectivity NWS

Korach and Stern presented an $\mathcal{O}(c^4 n^2)$-time algorithm for Connectivity NWS where $G$ is a clique and $t = 0$ [29] which was improved by Klemz et al. [28] who provided an $\mathcal{O}(m \cdot (c + \log(n)))$-time algorithm for Connectivity NWS with $t = 0$. Guttmann-Beck et al. [21] presented a similar algorithm for Unweighted Connectivity NWS with $t = 0$.

Next, we show that the positive result for $t = 0$ cannot be lifted to $t = 1$; in this case Connectivity NWS is NP-hard. We obtain our result by reducing from the NP-hard Hamiltonian Cycle-problem [1, 18], which asks for a given graph $G = (V, E)$ if there is a *Hamiltonian cycle* in $G$, that is, a cycle containing each vertex of $G$ exactly once.

▶ **Theorem 4.9** ($\star$). *Let $\Pi$ be a graph class on which Hamiltonian Cycle is* NP*-hard, then Unweighted Connectivity NWS is* NP*-complete on $\Pi$ even if $t = 1$.*

**Proof.** Let $I := (V, E)$ be an instance of Hamiltonian Cycle containing at least three vertices. We obtain an equivalent instance $I' := (G = (V, E), \mathcal{C}, \ell)$ of Unweighted Connectivity NWS as follows: We start with an empty set $\mathcal{C}$ and add for each vertex $v \in V$ a community $C_v := V \setminus \{v\}$ to $\mathcal{C}$. Finally, we set $\ell := |V|$. Note that $t = \ell - n + x$, where $x$ is the number of connected components of the graph. Thus, $t = n - n + 1 = 1$.

The detailed correctness proof is deferred to the full version. ◀

▶ **Corollary 4.10.** *Unweighted Connectivity NWS is NP-complete even if $t = 1$ on subcubic bipartite planar graphs.*

# 5 Stars NWS Parameterized by the Number of Communities

Unweighted Connectivity NWS is NP-hard even for $c = 7$ [17, Proposition 3]. In contrast, Stars NWS admits an XP-algorithm for $c$ with running time $n^{\mathcal{O}(c)}$: For each community $C$, test each of the at most $|C| \leq n$ potential center vertices. Then, for each potential solution check whether it consists of at most $\ell$ edges of total weight at most $b$. For

Stars NWS, we show that it is unlikely that this brute-force algorithm can be improved, by showing W[1]-hardness. For Unweighted Stars NWS, we obtain an FPT-algorithm for $c$.

▶ **Theorem 5.1** (⋆). *Stars NWS is* W[1]*-hard when parameterized by $c$ even if $G$ is a clique and each edge weight is* 1 *or* 2*.*

**Proof.** We provide a parameter-preserving reduction from the W[1]-hard Regular Multicolored Clique problem [11]. The input consists of an $r$-regular graph $G$, an integer $\kappa$, and a partition $(V_1, \dots V_\kappa)$ of $V(G)$. The question is whether there exists a clique of size $\kappa$ containing exactly one vertex of each partite set $V_i$.

We construct an equivalent instance $(G', \mathcal{C}, \omega, \ell, b)$ of Stars NWS as follows. The vertex set of $V(G')$ consists of a copy of $V(G)$ and $\kappa$ additional vertex sets $S_i$, $i \in [\kappa]$, each of size $n(G)^3$. We make $G'$ a clique by adding all edges between vertices of $V(G')$. To complete the construction, we specify the communities and edge weights. First, for each color class $i \in [1, \kappa]$, we add a community $C_i := V(G) \cup S_i$. Afterwards, we define the edge weights: For each edge $\{a, b\} \in E(G')$ such that $\{a, b\} \in E(G)$, we set $\omega(\{a, b\}) := 2$, for each edge $\{a, b\}$ with $a \in S_i$ and $b \notin V_i$, we set $\omega(\{a, b\}) := 2$, for each edge $\{a, b\}$ with $a \in S_i$ and $b \in S_j$, we set $\omega(\{a, b\}) := 2$, and for each remaining edge $\{a, b\} \in E(G')$ we set $\omega(\{a, b\}) := 1$. Finally, we set $\ell := \kappa \cdot (n(G)^3 + n(G) - 1) - \binom{\kappa}{2}$ and $b := \kappa \cdot (n(G)^3 + n(G) - 1 + r) - 2\binom{\kappa}{2}$. Note that $c = \kappa$, it thus remains to show the equivalence of the two instances. The detailed correctness proof is deferred to the full version. ◀

▶ **Theorem 5.2** (⋆). *Unweighted Stars NWS is solvable in $\mathcal{O}(4^{c^2} \cdot (n + m) + n^2 \cdot c)$ time.*

To complete the parameterized complexity picture, we show that a polynomial kernel for $c$ is unlikely.

▶ **Theorem 5.3** (⋆). *Unweighted Stars NWS parameterized by $c$ does not admit a polynomial kernel unless NP ⊆ coNP/poly.*

# 6 Conclusion

Presumably the most interesting open question is whether Stars NWS parameterized by $t$ admits an FPT-algorithm. In Theorem 4.2 we showed that Stars NWS can be solved in polynomial time if for some optimal solution the edge set of all local cycles is known. Thus, to obtain an FPT-algorithm, it is sufficient to find such an edge set in FPT-time. Also, it is open whether Unweighted Connectivity NWS can be solved in polynomial time when $t$ is constant and the input graph is a clique. In other words, it is open whether a minimum-edge hypergraph support can be found in polynomial time when it has a constant feedback edge number.

It is also interesting to close the gap between the running time lower bound of $2^{\Omega(c)} \cdot \mathrm{poly}(|I|)$ (see Proposition 2.5) and the upper bound of $2^{\mathcal{O}(c^2)} \cdot \mathrm{poly}(|I|)$ (see Theorem 5.2) for Unweighted Stars NWS. Also, we may ask the following: are there properties $\Pi$ such that $\Pi$-NWS is NP-hard but can be solved in $2^{\mathcal{O}(n)} \cdot \mathrm{poly}(n + c)$ time?

## References

1 Takanori Akiyama, Takao Nishizeki, and Nobuji Saito. NP-completeness of the Hamiltonian cycle problem for bipartite graphs. *Journal of Information Processing*, 3(2):73–76, 1980.

2 Dana Angluin, James Aspnes, and Lev Reyzin. Network construction with subgraph connectivity constraints. *Journal of Combinatorial Optimization*, 29(2):418–432, 2015.

3 Catriel Beeri, Ronald Fagin, David Maier, and Mihalis Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM*, 30(3):479–513, 1983.

4 Ulrik Brandes, Sabine Cornelsen, Barbara Pampel, and Arnaud Sallaberry. Path-based supports for hypergraphs. *Journal of Discrete Algorithms*, 14:248–261, 2012.

5 Kevin Buchin, Marc J. van Kreveld, Henk Meijer, Bettina Speckmann, and Kevin Verbeek. On planar supports for hypergraphs. *Journal of Graph Algorithms and Applications*, 15(4):533–549, 2011.

6 Chandra Chekuri, Thapanapong Rukkanchanunt, and Chao Xu. On element-connectivity preserving graph simplification. In *Proceedings of the 23rd Annual European Symposium on Algorithms (ESA '15)*, volume 9294 of *Lecture Notes in Computer Science*, pages 313–324. Springer, 2015.

7 Jiehua Chen, Christian Komusiewicz, Rolf Niedermeier, Manuel Sorge, Ondrej Suchy, and Mathias Weller. Polynomial-time data reduction for the subset interconnection design problem. *SIAM Journal on Discrete Mathematics*, 29(1):1–25, 2015.

8 Gregory V. Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Constructing scalable overlays for pub-sub with many topics. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing (PODC '07)*, pages 109–118. ACM, 2007.

9 Nathann Cohen, Frédéric Havet, Dorian Mazauric, Ignasi Sau, and Rémi Watrigant. Complexity dichotomies for the minimum $F$-overlay problem. *Journal of Discrete Algorithms*, 52-53:133–142, 2018.

10 Vincent Conitzer, Jonathan Derryberry, and Tuomas Sandholm. Combinatorial auctions with structured item graphs. In Deborah L. McGuinness and George Ferguson, editors, *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI '04)*, pages 212–218. AAAI Press / The MIT Press, 2004.

11 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms.* Springer, 2015.

12 Rodney G. Downey and Michael Ralph Fellows. *Fundamentals of Parameterized Complexity.* Springer Science & Business Media, 2013.

13 Ding-Zhu Du and Zevi Miller. Matroids and subset interconnection design. *SIAM Journal on Discrete Mathematics*, 1(4):416–424, 1988.

14 Pierre Duchet. Propriete de Helly et problemes de representation. *Colloque International CNRS, Problemes Conbinatoires et Theorie du Graphs*, 260:117–118, 1978.

15 Hongbing Fan, Christian Hundt, Yu-Liang Wu, and Jason Ernst. Algorithms and implementation for interconnection graph problem. In *Proceedings of the Second International Conference on Combinatorial Optimization and Applications (COCOA '08)*, volume 5165 of *Lecture Notes in Computer Science*, pages 201–210. Springer, 2008.

16 Claude Flament. Hypergraphes arborés. *Discrete Mathematics*, 21(3):223–227, 1978.

17 Till Fluschnik and Leon Kellerhals. Placing green bridges optimally, with a multivariate analysis. *Theory of Computing Systems*, 2024. To appear.

18 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

19 Aristides Gionis, Polina Rozenshtein, Nikolaj Tatti, and Evimaria Terzi. Community-aware network sparsification. In *Proceedings of the 2017 SIAM International Conference on Data Mining (SDM '17)*, pages 426–434. SIAM, 2017.

20 Nili Guttmann-Beck, Roni Rozen, and Michal Stern. Vertices removal for feasibility of clustered spanning trees. *Discrete Applied Mathematics*, 296:68–84, 2021.

21 Nili Guttmann-Beck, Zeev Sorek, and Michal Stern. Clustered spanning tree - conditions for feasibility. *Discrete Mathematics & Theoretical Computer Science*, 21(1), 2019.

22 Maike Herkenrath, Till Fluschnik, Francesco Grothe, and Leon Kellerhals. Placing green bridges optimally, with habitats inducing cycles. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, (IJCAI '22)*, pages 3825–3831. ijcai.org, 2022.

23 Emanuel Herrendorf. On the complexity of community-aware network sparsification. Master's thesis, Philipps-Universität Marburg, 2022. URL: `https://www.fmi.uni-jena.de/fmi_femedia/24200/master-emanuel-pdf.pdf`.

24 Jun Hosoda, Juraj Hromkovic, Taisuke Izumi, Hirotaka Ono, Monika Steinová, and Koichi Wada. On the approximability and hardness of minimum topic connected overlay and its special instances. *Theoretical Computer Science*, 429:144–154, 2012.

25 Russell Impagliazzo and Ramamohan Paturi. On the complexity of $k$-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

26 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

27 David S. Johnson and Henry O. Pollak. Hypergraph planarity and the complexity of drawing venn diagrams. *Journal of Graph Theory*, 11(3):309–325, 1987.

28 Boris Klemz, Tamara Mchedlidze, and Martin Nöllenburg. Minimum tree supports for hypergraphs and low-concurrency Euler diagrams. In *Proceedings of the 14th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT '14)*, volume 8503 of *Lecture Notes in Computer Science*, pages 265–276. Springer, 2014.

29 Ephraim Korach and Michal Stern. The clustering matroid and the optimal clustering tree. *Mathematical Programming*, 98(1-3):385–414, 2003.

30 Ephraim Korach and Michal Stern. The complete optimal stars-clustering-tree problem. *Discrete Applied Mathematics*, 156(4):444–450, 2008.

31 Gerd Lindner, Christian L. Staudt, Michael Hamann, Henning Meyerhenke, and Dorothea Wagner. Structure-preserving sparsification of social networks. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '15)*, pages 448–454. ACM, 2015.

32 Natsu Nakajima, Morihiro Hayashida, Jesper Jansson, Osamu Maruyama, and Tatsuya Akutsu. Determining the minimum number of protein-protein interactions required to support known protein complexes. *PloS one*, 13(4):e0195545, 2018.

33 Peter J Slater. A characterization of soft hypergraphs. *Canadian Mathematical Bulletin*, 21(3):335–337, 1978.

34 Robert Endre Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984.

35 Fang Zhou, Sébastien Mahler, and Hannu Toivonen. Network simplification with minimal loss of connectivity. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM '10)*, pages 659–668. IEEE Computer Society, 2010.