Algorithmic Aspects of Hypergraph Connectivity Augmentation and Graph Density Deletion

Shubhang Kulkarni

Contents

1 Introduction

The concept of a graph being *well-connected* is often linked to how closely the graph resembles the complete graph. This is because, like the complete graph, a well-connected graph should resist being split into disjoint parts on omission of a small number of edges. However, designing and maintaining large graphs that mirror the structure of the complete graph is impractical due to the excessive number of direct connections required. For example, in several applications where networks are modeled using graphs, each vertex is responsible for storing information about its incident edges. In a *dense* graph, this would mean a single vertex must store data about an overwhelming number of edges, leading to significant storage and maintenance costs. Thus, there exists a close relation between the *connectivity* and *density* of a graph. High connectivity requirement is often a natural goal in applications, but it typically comes with increased density, which may be costly. In this proposal, we present our progress in the study of these two properties—we present published results, ongoing work, and open problems for future research. In particular, we focus on the following fundamental question in network design and optimization.

Question 1. *What is the best way to update a given network to achieve a (high) target connectivity or (low) target density?*

We note that addressing Question 1 begins with specifying its two abstract components—the *network model* and the permissible *updates* to the network. The network models we focus on are *graphs* and *hypergraphs*—graphs are the most natural and intuitive way to represent networks involving pairwise interactions, whereas hypergraphs naturally model more complex relationships such as multi-way interactions that cannot be captured by traditional graph edges. The types of network updates that we consider categorize the problems presented in the proposal into three broad categories:

- **Deletion Problems (for Density Reduction).** We consider the operation of deleting endpoints (and their incident connections) to reduce network density.
- **Augmentation Problems (for Connectivity Enhancement).** We consider the operation of adding new connections to enhance network connectivity.
- **Splitting-Off or Connection-Reconfiguration (for Connectivity Preservation).** We consider the operation of reconfiguring connections amongst endpoints in order to simplify the network without loss in connectivity.

We primarily focus on deletion problems in *graphs*, and augmentation and splitting-off problems in *hypergraphs* as our main results. We then go beyond concrete network models and consider these problems in the more abstract regime of *submodular and supermodular functions*. Submodularity is a unifying theme across all results we will discuss in this proposal—this theme will become more apparent as we delve further into the technical sections. We now describe some background and notation, and then give an overview of the specific questions of interest in the rest of the proposal.

Preliminaries. A hypergraph $G = (V, E)$ consists of a finite set V of vertices and a set E of hyperedges, where every hyperedge $e \in E$ is a subset of V. We will denote a hypergraph $G = (V, E)$ with hyperedge weights $w : E \to \mathbb{Z}_+$ by the tuple (G, w) and use G_w to denote the unweighted multi-hypergraph over vertex set V containing $w(e)$ copies of every hyperedge $e \in E$. Throughout this work, we will be interested only in hypergraphs with positive integral weights and for algorithmic problems where the input/output is a hypergraph, we will require that the weights are represented in binary. If all hyperedges have size at most 2, then the hyperedges are known as edges and we call such a hypergraph as a graph.

Notation. Let $(G = (V, E), w : E \to \mathbb{Z}_+)$ be a hypergraph. For $X \subseteq V$, let $\delta_G(X) := \{e \in E : e \cap X \neq \emptyset, e \setminus X \neq \emptyset\}$. We define the cut function $d_{(G,w)} : 2^V \to \mathbb{Z}_{\geq 0}$ by $d_{(G,w)}(X) := \sum_{e \in \delta_G(X)} w(e)$ for every $X \subseteq V$. For a vertex $v \in V$, we use $d_{(G,w)}(v)$ to denote $d_{(G,w)}(\{v\})$. We define the degree of a vertex v to be the sum of the weights of hyperedges containing v—we note that the degree of a vertex is not necessarily equal to $d_{(G,w)}(v)$ since we could have $\{v\}$ itself as a hyperedge (i.e., a singleton hyperedge that contains only the vertex v). For distinct vertices $u, v \in V$, let $\lambda_{(G,w)}(u,v) \coloneqq \min\{d_{(G,w)}(X) : u \in X \subseteq V \setminus \{v\}\}$ – i.e., $\lambda_{(G,w)}(u,v)$ is the value of a minimum $\{u,v\}$-cut in the hypergraph. If all hyperedge weights are unit, then we drop w from the subscript and simply use d_G and λ_G.

1.1 Splitting-off in Hypergraphs

The splitting-off operation in undirected graphs is a simple yet powerful operation in graph theory. It is a reduction operation that enables a vertex to exit the network by informing its neighbors how to reconfigure the lost links among themselves in order to preserve their degrees. Lovász [84, 86] introduced the splitting-off operation and showed the existence of the operation to preserve global edge-connectivity under certain conditions. Mader [87] showed the existence of the splitting-off operation to preserve local edge-connectivity (i.e., all pairwise edge-connectivities) under certain conditions. Both Lovász's and Mader's results also admit strongly polynomial-time algorithms [50, 53, 90]. Owing to the inductive nature of the splitting-off operation, Lovász's and Mader's results have enabled fundamental results in graph theory as well as efficient algorithms and min-max relations for numerous graph optimization problems. In fact, Mader [87] illustrated the power of his local edge-connectivity preserving splitting-off result by deriving Nash-Williams' strong orientation theorem [94] (also see Frank's exposition of this derivation [48]). Subsequently, the splitting-off operation has been used to give constructive characterizations of k-edge-connected and k-tree-connected graphs [50] and to address problems in edge-connectivity augmentation [4, 47, 49, 50, 90], graph orientation [51, 74], minimum cuts enumeration [56, 65, 91], network design [27, 36, 57, 70, 85], tree packing [19, 68, 76, 78], congruency-constrained cuts [92], and approximation algorithms for TSP [20, 57]. Designing fast algorithms for global/local edge-connectivity preserving splitting-off remains an active area of research (e.g., see recent works [24, 25, 26, 79]) due to these far-reaching applications.

Hypergraphs offer a richer and more accurate model than graphs for several applications. Consequently, hypergraphs have found applications in several modern areas (e.g., see [82, 96, 102, 111]) and these applications have, in turn, driven exciting recent progress in algorithms for hypergraph optimization problems [2, 7, 28, 29, 31, 37, 38, 46, 54, 60, 69, 71, 72, 75, 80, 99, 104]. The far reaching implications of the splitting-off operation in graph theory and optimization naturally raise the following question.

Question 2. *Is there an analogue of the splitting-off operation in hypergraphs?*

We note that Question 2 asks for a *new* operation on hypergraphs that is degree-preserving and allows a vertex to leave the hypergraph on repeated application of the operation—such an operation has not appeared in literature before. In Section 2, we answer Question 2 by defining such an operation (see Definition 2.1). We note that the far-reaching applications of splitting-off arise because of the existence of local connectivity-preserving splitting-off in graphs by Mader's Theorem. This raises the following question.

Question 3. *Does there exist a local connectivity-preserving complete splitting-off in hypergraphs?*

One of our main result (see Theorem 2.1) answers Question 3 by showing an analogue of Mader's Theorem for our definition of splitting-off in hypergraphs. We then turn our attention to the algorithmic aspects of computing a local-connectivity preserving *complete* splitting-off in hypergraphs, i.e. efficiently computing a hypergraph obtained by repeated local-connectivity preserving splitting-off operations to isolate a particular vertex. For this, we specify the representation of the input hypergraph—we consider *weighted* hypergraphs representing multi-hypergraphs, where all hyperedges are distinct and the weight of a hyperedge (represented in binary as part of the input) indicates the number of copies of the hyperedge present in the hypergraph. Then, the following question summarizes the algorithmic challenge.

Question 4. *Can a local connectivity-preserving complete splitting-off in weighted hypergaphs be computed in strongly polynomial time?*

Our main result (see Theorem 2.1) also answers Question 4 affirmatively. Finally, due to the widespread applications of splitting off in graphs, it is natural to wonder about the applicability of the new splitting-off operation for hypergraphs.

Question 5. *Are there interesting applications of the local-connectivity preserving splitting-off operation in hypergraphs?*

We answer Question 5 by showing two applications of our local-connectivity preserving splitting-off operation. First, we show that the operation can be used to give a constructive characterization of k-hyperedge-connected hypergraphs (see Section 2.3.2). We also give an alternate proof of an approximate min-max result for Steiner rooted connected orientations of hypergraphs by Kiraly and Lau [74] (see Section 2.3.1). As a special case, we obtain a simple proof of Menger's Theorem.

Extensions to symmetric skew-supermodular functions. Several results on hypergraph connectivity are actually special cases of general results involving submodular functions—see [50] for several such examples in literature. In Section 2.4, we mention how this is also the case for our splitting-off result. In particular, we consider the algorithmic problem of systematically converting a *weak cover* of a *symmetric skew-supermodular* function to a *strong cover* of the function. The result presented in this section (see Theorem 2.5) captures our local-connectivity preserving splitting-off as a special case.

1.2 Degree-Specified Hypergraph Connectivity Augmentation.

A fundamental problem in graph optimization is the problem of augmenting a given graph using edges so that it satisfies target connectivity requirements and degree specifications—formally defined below.

Definition 1.1 (DS-Graph-CA-using-E)**.** Degree-specified Graph Connectivity Augmentation using Edges problem *is defined as follows:*

Input:	*A graph* $(G = (V, E_G, c_G : E_G \to \mathbb{Z}_+)$, *target connectivity function* $r : \binom{V}{2} \to \mathbb{Z}_{\geq 0}$, *and* *degree requirement function* $m : V \to \mathbb{Z}_{\geq 0}$.
Goal:	*Verify if there exists a graph* $(H = (V, E_H), w_H : E_H \to \mathbb{Z}_+)$ *such that* *(1)* $b_{(H,w_H)}(u) = m(u)$ *for every* $u \in V$, *(2)* $\lambda_{(G+H,c_G+w_H)}(u, v) \geq r(u, v)$ *for every distinct* $u, v \in V$, *and if so, then find such a graph.*

Watanabe and Nakamura [112] introduced DS-Graph-CA-using-E for the case of uniform requirement function (i.e., $r(u, v) = k$ for all distinct $u, v \in V$ for some $k \in \mathbb{Z}_+$) and showed that this case is solvable in polynomial time in unweighted graphs. Subsequently, Frank [47] gave a strongly polynomial-time algorithm for DS-Graph-CA-using-E. Since then, designing fast algorithms as well as parallel algorithms for DS-Graph-CA-using-E has been an active area of research [11, 12, 19, 23, 49, 53, 79, 93]. The last couple of years has seen exciting progress for the uniform requirement function culminating in a near-linear time algorithm [24, 25, 26]. In addition to making progress in the algorithmic status of the problem, these works have revealed fundamental structural properties of graph cuts which are of independent interest in graph theory.

Remark 1.1. *We note that DS-Graph-CA-using-E is a feasibility problem. There is a closely related optimization variant where the goal is to find the appropriate augmenting graph* $(H = (V, E_H), w_H : E_H \to \mathbb{Z}_+)$ *with minimum total weight* $\sum_{e \in E_H} w_H(e)$. *This optimization version is different from the NP-hard min-cost connectivity augmentation problems (like Steiner tree and tree/cactus/forest augmentation) whose approximability have been improved recently [21, 59, 107, 108, 109]. All algorithms to solve the optimization version [11, 12, 19, 23, 47, 49, 53, 79, 93, 112] reduce it to solving the degree-specified feasibility version and so we focus on just this version in this proposal.*

We consider the following extension of DS-Graph-CA-using-E to hypergraphs.

Definition 1.2 (DS-Hypergraph-CA-using-H)**.** Degree-specified Hypergraph Connectivity Augmentation using Hyperedges problem *is defined as follows:*

Input:	*A hypergraph* $(G = (V, E_G, c_G : E_G \to \mathbb{Z}_+)$, *target connectivity function* $r : \binom{V}{2} \to \mathbb{Z}_{\geq 0}$, *and* *degree requirement function* $m : V \to \mathbb{Z}_{\geq 0}$.
Goal:	*Verify if there exists a hypergraph* $(H = (V, E_H), w_H : E_H \to \mathbb{Z}_+)$ *such that* *(1)* $b_{(H,w_H)}(u) = m(u)$ *for every* $u \in V$, *(2)* $\lambda_{(G+H,c_G+w_H)}(u, v) \geq r(u, v)$ *for every distinct* $u, v \in V$, *and if so, then find such a hypergraph.*

Szigeti [106] showed that DS-Hypergraph-CA-using-H can be solved in pseudo-polynomial time. In particular, if the target connectivity function $r : \binom{V}{2} \to \mathbb{Z}_{\geq 0}$ is given in unary, then the problem can be solved in polynomial time. Since DS-Graph-CA-using-E can be solved in strongly polytime algorithm, we arrive at the following natural question.

Question 6. *Does there exist a strongly polynomial time algorithm for DS-Hypergraph-CA-using-H?*

Attempting to answer Question 6 brings up a subtle point which contrasts graph and hypergraph network design. It is clear that DS-Graph-CA-using-E is in NP since a YES instance admits a weighted *graph* (H, w_H) as a feasible solution which serves as a polynomial-time verifiable certificate for the YES instance; in contrast, it is not immediately clear whether DS-Hypergraph-CA-using-H is in NP. This is because, the number of hyperedges in the desired hypergraph (H, w_H) could be exponential in the number of vertices, and consequently, exponential in the size of the input. In fact, Szigeti's result also implies that if the input instance is feasible, then it admits a solution hypergraph (H, w_H) such that the number of hyperedges in H is at most $\max\{2^{|V|}, \max\{r(u, v) : \{u, v\} \in \binom{V}{2}\}\}$. Thus, to answer the algorithmic Question 6, we would also need to answer the following weaker structural question as a necessary step.

Question 7. *For a YES instance of DS-Hypergraph-CA-using-H, does there exist a solution hypergraph with (strongly) polynomial number of distinct hyperedges?*

In Section 3, we answer both Question 7 and Question 6 in the affirmative (see Theorem 3.1). We also mention how these results can be extended to obtain solution hypergraphs with additional properties. A hypergraph is *uniform* if all hyperedges have the same size; a hypergraph is *near-uniform* if every pair of hyperedges differ in size by at most one. Uniformity/near-uniformity is a natural constraint in network design applications involving hypergraphs—we might be able to create only equal-sized hyperedges in certain applications. Requiring uniform (or near-uniform) hyperedges can also be viewed as a fairness inducing constraint in certain applications. We consider the natural variant of DS-Hypergraph-CA-using-H under near-uniformity constraints and show that we can compute a solution hypergraph in strongly polynomial time that has linear number of hyperedges and is *near-uniform* (see Theorem 3.2) or one that has quadratic number of hyperedges, is near-uniform, and simultaneously augments the connectivity of two input hypergraphs (see Theorem 3.3). Our results are extensions of results by Bernáth and Király [17] who showed the existence of near-uniform solution hypergraphs with pseudo-polynomial number of distinct hyperedges.

Extensions to skew-supermodular functions. Similar to the previous section, we show that our results mentioned above can be extended to *skew-supermodular* functions. In particular, in Section 3.2, we consider the algorithmic problem of computing *weak covers* of *skew-supermodular functions*. The results (Theorem 3.5 and Theorem 3.6) mentioned above are corollaries to the result presented in this section (and the results of Section 2). In Section 3.3, we mention some applications of our general result to *degree-specified node-to-area hypergraph connectivity augmentation* (see Theorem 3.7) and *degree-constrained mixed hypergraph connectivity augmentation* (See Theorem 3.8) using hyperedges.

Remark 1.2. *All the results that we present in Section 3 can be extended to appropriate optimization variants which are similar to the variant mentioned in Remark 1.1 for graphs. We avoid stating those variants and corresponding results for the sake of brevity.*

1.3 Density Deletion Set

The *densest subgraph* problem in graphs (DSG) is a core primitive in graph and network mining applications [55] . In DSG, we are given a graph $G = (V, E)$ and the goal is to find $\lambda^*_G := \max_{S \subseteq V} |E(S)|/|S|$, where $E(S)$ is the set of edges with both end vertices in S. DSG can be computed in polynomial time [32,58,98], and thus it is of central importance in various applications [77]. In fact, DSG is a fundamental problem in algorithms and combinatorial optimization with several connections graph theory, matroids, and submodularity. There are many recent papers exploring various aspects of DSG and related problems [35, 62, 73, 100, 105, 110]. Due to the widespread significance of DSG, a natural line of research is to study robustness of DSG under perturbations to the graph. For example, the classic work of Cunningham on the attack problem [42] can be seen as addressing the robustness of DSG by examining how it behaves under *edge deletions*. In certain parameter regimes, this edge-deletion problem can be solved in polynomial time using matroid and network flow techniques. Here, we will be interested in the natural *vertex-deletion* variant. We formally define this problem next.

Definition 1.3 (GraphDD)**.** *The problem is defined as follows:*

Input:	*Graph* $G = (V, E)$,
	Vertex deletion costs $c : V \to \mathbb{R}_+$, *and*
	Positive integer $\rho \in \mathbb{Z}_+$.
Goal:	*Compute* $\arg\min\{c(F) : F \subseteq V$ *and density*$(G - F) \leq \rho\}$.

When the target density ρ is fixed (i.e. a constant), we denote the problem as ρ-GraphDD . We observe that 0-GraphDD is equivalent to the vertex cover problem since requiring density of 0 after deleting S is equivalent to S being a vertex cover of G. One can also see, in a similar fashion, that 1-GraphDD is equivalent to the pseudoforest deletion set problem, denoted PFDS (where the goal is to delete vertices so that every component in the remaining graph has at most one cycle), and $(1-1/|V|)$-GraphDD is equivalent to the feedback vertex set problem, denoted by FVS (where the goal is to delete vertices so that the remaining graph is acyclic). While ρ-GraphDD is natural in its formulation, to the best of our knowledge, GraphDD has only recently been explicitly defined and explored by Bazgan, Nichterlein and Vazquez Alferez [9] from an FPT perspective. A natural question is to understand the approximability of the problem. Towards this, Fujito [52] studied the problem of MatroidFVS which problem captures ρ-GraphDD for $\rho \in \mathbb{Z}_+$ as a special case. Fujito showed that this problem admits a $O(\log n)$ approximation. Since the only known approximation lower bound for GraphDD is $(2-\epsilon)$-inapproximability under the Unique Games Conjecture, (arising from that of VertexCover), resolving the gap in the approximation status of the problem is a natural question.

Question 8. *What is the approximability of the ρ-GraphDD problem?*

In Section 4.1 we answer Question 8 by showing that ρ-GraphDD is $o(\log n)$-inapproximable unless P=NP (see Theorem 4.1). We note that this is particularly surprising since the special cases of $\rho \leq 1$ admit 2-approximation algorithms—we will come back to these special cases below. Thus, the problem admits a *phase transition* in its approximability at $\rho = 1$. In order to overcome the inapproximability of the problem we ask the next algorithmic question which allows relaxing the target density requirement.

Question 9. *Does there exist a bicriteria-approximation algorithm for GraphDD?*

In Section 4.1, we answer Question 9 in the affirmative (see Theorem 4.2). Our bicriteria algorithm is based on a new LP formulation for the ρ-GraphDD problem. This formulation arose due to insights we obtained from re-examining the special case of PFDS from the viewpoint of obtaining alternative LP formulations; and this was inspired from re-examining FVS from a similar viewpoint. The next two subsections are devoted to describing our new results and insights for these two special cases—in Section 1.3.1 we focus on FVS and in Section 1.3.2 we focus on PFDS. Before delving into these special cases, we conclude this section by briefly mentioning ongoing work which extends the results to the abstract setting of supermodular functions.

Extensions to normalized, monotone supermodular functions. The *density* of a *supermodular* function $f : 2^V \to \mathbb{R}$ is defined as density$(f) := \max_{S \subseteq V} \frac{f(S)}{|S|}$. Given a graph $G = (V, E)$, we note that considering the supermodular function $f(S) := |E[S]|$ for all $S \subseteq V$ recovers the notion of graph density considered earlier in the section. In Section 4.2 we describe the natural generalization of the GraphDD problem for (normalized, monotone) supermodular functions, namely SupmodDD. Since GraphDD is a special case, our inapproximability result also carries over to the general setting. A natural question here is to obtain a bicriteria algorithm for SupmodDD analogous to our answer to Question 9. In Section 4.2, we present the result (see Theorem 4.4) of an ongoing work that answers this question in the affirmative by giving a *randomized combinatorial* algorithm. Interestingly, the approximation guarantees that we are able to obtain depend on a parameter c_f of the input supermodular function f which was recently introduced by Chekuri, Quanrud and Torres [35] to unify the analysis of the greedy peeling algorithm for variants of the densest subgraph and extensions to supermodular density.

1.3.1 Feedback Vertex Set

FVS is a fundamental vertex deletion problem in the field of combinatorial optimization and appears in Karp's list of 21 NP-Complete problems. For a graph $G = (V, E)$, a subset $U \subseteq V$ is a *feedback vertex set* if $G - U$ is acyclic; in other words U is a hitting set for the cycles of G. We formally define the optimization problem below.

Definition 1.4 (FVS)**.** *The problem is defined as follows:*

Input:	*Graph* $G = (V, E)$*, and* *Vertex deletion costs* $c : V \to \mathbb{R}_+$
Goal:	*Compute* $\arg\min \left\{ \sum_{u \in U} c(u) : U \subseteq V \text{ and } U \text{ is a feedback vertex set for } G \right\}$.

Approximation algorithms for the unweighted version of FVS (UFVS), i.e., when all vertex costs are one, have been studied since 1980s. An $O(\log n)$-approximation for UFVS is implicit in the work of Erdös and Pósa [43]; Monien and Schulz [89] seem to be the first ones to explicitly study the problem, and obtained an $O(\sqrt{\log n})$-approximation for UFVS. Bar-Yehuda, Geiger, Naor, and Roth [8] improved the ratio for UFVS to 4, and also noted that an $O(\log n)$-approximation holds for FVS. Soon after that, Bafna, Berman, and Fujito [3], and independently Becker and Geiger [10], obtained 2-approximation algorithms for FVS. While [3] explicitly uses the local-ratio terminology, [10] describes the algorithm in a purely combinatorial fashion. Although the algorithm in [3] is described via the local-ratio method, the underlying LP relaxation is not obvious. As observed in [8], the natural hitting set LP relaxation for FVS has an integrality gap of $\Theta(\log n)$. Chudak, Goemans, Hochbaum, and Williamson [39] described exponential sized integer linear programming (ILP) formulations for FVS, and showed that the algorithms in [3,10] can be viewed as primal-dual algorithms with respect to the LP relaxations of these formulations. This also established that these LP relaxations have an integrality gap of at most 2. As we mentioned at the beginning of the main section, Fujito [52] considered a unified generalization of FVS and vertex cover, namely matroidal FVS, and gave a $O(\log n)$-approximation for matroidal FVS via connections to submodular set cover. In the same work, Fujito formulated an exponential sized ILP for matroidal FVS, and designed a primal-dual algorithm with respect to its LP relaxation. He proved that the algorithm has an approximation ratio of 2 for a certain family of matroids. When specialized to FVS, we note that the ILP and the resulting primal-dual algorithm in [52] is slightly different from the one in [39] although they both yield 2-approximations.

Although the LPs of Fujito and Chudak et. al. have integrality gap at most 2 for FVS, it is not known whether these can actually be solved in polynomial time. Equivalently, it is open to design polynomial-time separation oracles for the families constraints of these LPs. Moreover, there has been no other polynomial-time solvable LP relaxation through which one could obtain a 2-approximation for FVS. The lack of solvable

LP relaxations for FVS with small integrality gap has also been a stumbling block for the design of approximation algorithms for a generalization of FVS called the subset feedback vertex set problem (Subset-FVS) [44]: the input is a vertex-weighted graph G and a terminal set $T \subseteq V$, and the goal is to remove a minimum cost subset of vertices S to ensure that $G - S$ has no cycle containing a terminal $t \in T$. There is an 8-approximation for Subset-FVS [45] and this is based on a complex algorithm that combines combinatorial and LP-based techniques. Chekuri and Madan [34] formulated a polynomial-sized integer linear program for Subset-FVS and showed that the integrality gap of its LP-relaxation is at most 13. They explicitly raised the question of whether the integrality gap of their formulation is better for FVS; in fact it is open whether their formulation's integrality gap is at most 2 for Subset-FVS. This brings us to the following motivating question.

Question 10. *Does there exist an ILP formulation for FVS whose LP-relaxation can be solved in polynomial time and has integrality gap at most* 2*?*

In Section 4, we affirmatively answer Question 10 (see Theorem 4.8). We also affirmatively answer the question posed by Chekuri and Madan by analyzing the integrality gap of their LP when specialized to FVS.

1.3.2 Pseudoforest Deletion Set

A connected graph is a *pseudotree* if it has exactly one cycle; in other words there is an edge whose removal results in a spanning tree. A graph is a *pseudoforest* if every connected component is either acyclic or a pseudotree. For a graph $G = (V, E)$, a subset $U \subseteq V$ is a *pseudoforest deletion set* if $G - U$ is a pseudoforest. We formally define the optimization problem below.

Definition 1.5 (PFDS)**.** *The problem is defined as follows:*

Input: *Graph* $G = (V, E)$*, and*
Vertex deletion costs $c : V \to \mathbb{R}_+$

Goal: *Compute* $\arg\min \left\{ \sum_{u \in U} c(u) : U \subseteq V \text{ and } U \text{ is a pseudoforest deletion set for } G \right\}$.

Intuitively, one can see that PFDS is closely related to FVS: we note that a feasible solution for FVS in a given graph G is a feasible solution to the PFDS instance on G. Also, finding an FVS in a graph that is a pseudoforest is easy: for each connected component that is a pseudotree, we remove the cheapest vertex in its unique cycle. PFDS and FVS are special cases of the more general ℓ-pseudoforest deletion problem that was introduced in [97] from the perspective of parameterized algorithms (FVS corresponds to $\ell = 0$ and PFDS to $\ell = 1$). Lin, Feng, Fu, and Wang [83] studied approximation algorithms for ℓ-pseudoforest deletion problem. In this paper, we restrict attention to PFDS and FVS and do not discuss the more general ℓ-pseudoforest deletion problem. The status of PFDS is very similar to that of FVS. It has an approximation preserving reduction from the vertex cover problem and consequently, it is NP-hard, and does not have a polynomial time $(2 - \epsilon)$-approximation for every constant $\epsilon > 0$ assuming the UGC. It admits a polynomial-time 2-approximation based on the local-ratio technique [83]. While the authors in [83] do not explicitly discuss LP relaxations for PFDS, we note that their local-ratio algorithm can be converted to an LP-based 2-approximation for PFDS following the ideas in [39]. In fact, PFDS also falls under the matroidal FVS framework of Fujito for sparse matroids, and so Fujito's LP, when specialized for PFDS, also has an integrality gap of at most 2. However, similar to the situation for FVS, we do not know how to solve these LPs in polynomial time. This leads to the following question (which is the counterpart of Question 10 for PFDS):

Question 11. *Does there exist an ILP formulation for PFDS whose LP-relaxation can be solved in polynomial time and has integrality gap at most* 2*?*

In Section 4, we present our results that answer Question 11 in the affirmative by exploiting connections to graph density (see Theorem 4.5). As we will see, the LPs which we we will exhibit as our answer to

Question 11 will play a crucial role in answering Question 10 for FVS. In addition to answering Question 11, we also exhibit extreme point properties of the underlying polytopes (see Theorem 4.6 and Theorem 4.7) which may be of independent interest.

2 Splitting off in hypergraphs

In this section we present our results on splitting-off in hypergraphs which appeared in ICALP 2024 [15]. In Section 2.1, we introduce our definition of a new splitting-off operation in *hypergraphs*. In Section 2.2, we describe the existence of local-connectivity preserving splitting-off operation analogous to Mader's Theorem and show that it can be computed in strongly polynomial-time algorithm in weighted hypergraphs. In Section 2.3, we illustrate the usefulness of our splitting-off operation by giving two applications. In Section 2.4, we mention how we can view splitting-off in hypergraphs as a special case of an abstract function covering problem on skew-supermodular functions—our results for existence and algorithms for local-connectivity preserving splitting-off in hypergraphs extend to this general regime.

2.1 Splitting-off operation

In this section, we introduce our definition of splitting-off in hypergraphs. To compare and contrast our definition of splitting-off for hypergraphs with the classical definition of splitting-off for graphs, we include both our definition and the classical definition and distinguish them by identifying them as h-splitting-off and g-splitting-off.

Definition 2.1 ([15]). *Let $(G = (V, E), w : E \to \mathbb{Z}_+)$ be a hypergraph and $s \in V$.*

1. *In* merge almost-disjoint hyperedges*, we pick a pair of hyperedges $e, f \in \delta_G(s)$ such that $e \cap f = \{s\}$, pick a positive integer $\alpha \in \mathbb{Z}_+$ such that $\alpha \le \min\{w(e), w(f)\}$, reduce the weights of hyperedges e and f by α, and increase the weight of a hyperedge g by α. Here,*
   (a) *if we choose $g := e \cup f$, then the associated operation will be called as* h-merge almost-disjoint hyperedges operation.
   (b) *if we choose $g := (e \cup f) \setminus \{s\}$, then the associated operation will be called as* g-merge almost-disjoint hyperedges operation.

   In the above, if $\alpha = w(e)$ (resp. if $\alpha = w(f)$), then we discard the hyperedge e (resp. hyperedge f) from the hypergraph obtained after the operation; if the hyperedge $g \notin E$, then we introduce g as a new hyperedge with weight $w(g) := 0$ before performing the weight increase on the hyperedge g.

2. *In* trim hyperedge operation*, we pick a hyperedge $e \in \delta_G(s)$, pick a positive integer $\alpha \in \mathbb{Z}_+$, reduce the weight of the hyperedge e and increase the weight of the hyperedge $g := e \setminus \{s\}$. Here,*
   (a) *if we choose $\alpha \le w(e)$, reduce the weight of the hyperedge e by α, and increase the weight of the hyperedge g by α, then the associated operation will be called as* h-trim operation *(if $\alpha = w(e)$, then we discard e from the hypergraph obtained after the operation; if $g \notin E$, then we add g as a new hyperedge with weight $w(g) := 0$ before performing the weight increase on the hyperedge g).*
   (b) *if we choose $\alpha \le w(e)/2$, reduce the weight of the hyperedge e by 2α, and increase the weight of the hyperedge g by 2α, then the associated operation will be called as* g-trim operation *(if $\alpha = w(e)/2$, then we discard e from the hypergraph obtained after the operation; if $g \notin E$, then we add g as a new hyperedge with weight $w(g) := 0$ before performing the weight increase on the hyperedge g).*

3. *We say that a hypergraph $(H = (V, E_H), w_H : E_H \to \mathbb{Z}_+)$ is obtained by applying a*
   (a) h-splitting-off operation at s from (G, w) *if (H, w_H) is obtained from (G, w) by either the h-merge almost-disjoint hyperedges operation or the h-trim hyperedge operation.*

(b) g-splitting-off operation at s from (G, w) *if* (H, w_H) *is obtained from* (G, w) *by either the g-merge almost-disjoint hyperedges operation or the g-trim hyperedge operation.*

Certain remarks regarding the definitions are in order. Firstly, the trim operation is valuable and unique to hypergraphs. It has been used in the hypergraph literature to obtain small-sized certificates for hypergraph connectivity [37] and for certain notions of directed hypergraph connectivity [50]. Secondly, all operations mentioned above are degree preserving for vertices $u \in V \setminus \{s\}$: both h-trim and g-trim operations preserve degrees by definition; both h-merge and g-merge *almost-disjoint* hyperedges operations preserve degrees due to the almost-disjoint property of the chosen hyperedges. Thirdly, all operations mentioned above *do not* increase the cut values of subsets $X \subseteq V \setminus \{s\}$. Thus, the relevant goal with these operations is ensuring that the cut values do not decrease too much—i.e., preserving global/local connectivity. We will be interested in repeated application of h-splitting-off operations at a vertex from a given hypergraph to isolate that vertex while preserving global/local connectivity. We define these formally next.

Definition 2.2 ([15]). *Let* $(G = (V, E), w : E \to \mathbb{Z}_+)$ *be a hypergraph and* $s \in V$.

1. *We say that a hypergraph* $(G^* = (V, E^*), w^* : E^* \to \mathbb{Z}_+)$ *is a*
   (a) complete h-splitting-off at s from (G, w) *if* $d_{(G^*,w^*)}(s) = 0$ *and* (G^*, w^*) *is obtained from* (G, w) *by repeatedly applying h-splitting-off operations at* s *from the current hypergraph.*
   (b) complete g-splitting-off at s from (G, w) *if* $d_{(G^*,w^*)}(s) = 0$ *and* (G^*, w^*) *is obtained from* (G, w) *by repeatedly applying g-splitting-off operations at* s *from the current hypergraph.*
2. *Let* (G^*, w^*) *be a complete h-splitting-off/g-splitting-off at* s *from* (G, w). *We say that* (G^*, w^*)
   (a) *preserves local connectivity if* $\lambda_{(G^*,w^*)}(u, v) = \lambda_{(G,w)}(u, v)$ *for every distinct* $u, v \in V \setminus \{s\}$ *and*
   (b) *preserves global connectivity if*
   $\min\{\lambda_{(G^*,w^*)}(u, v) : u, v \in V \setminus \{s\}, u \neq v\} = \min\{\lambda_{(G,w)}(u, v) : u, v \in V \setminus \{s\}, u \neq v\}$.

A notion of hypergraph splitting-off motivated by hypergraph connectivity augmentation applications has been studied in the literature before [6, 18, 40]. These works have explored local connectivity preserving complete g-splitting-off at a vertex s from a hypergraph under the assumption that *all hyperedges incident to the vertex* s *are edges (i.e., have size at most* 2*)*. In contrast, our focus is on local connectivity preserving complete h-splitting-off at a vertex s from a hypergraph without any assumption on the size of the hyperedges incident to the vertex s (i.e., the vertex s could have arbitrary-sized hyperedges incident to it). See [15] for an example of complete h-splitting-off at a vertex from a hypergraph.

We will primarily be concerned with complete *h-splitting-off* at a vertex from a *hypergraph* and complete *g-splitting-off* at a vertex from a *graph*. Complete g-splitting-off at a vertex from a *graph* is equivalent to the classical and well-studied notion of complete splitting-off at a vertex from a graph (for the definition of the classical notion in graphs, see [50, 90]). We cast the results of Lovász [84, 86] and Mader [87] in the framework of our definitions now. Let (G, w) be a *graph* and let s be a vertex in G. Lovász [84, 86] showed that if $d_{(G,w)}(\{s\})$ is even and $\min\{\lambda_{(G,w)}(u, v) : u, v \in V \setminus \{s\}\} \geq K$ for some $K \geq 2$, then there exists a *global* connectivity preserving complete g-splitting-off at the vertex s from (G, w). Mader [87] showed that if $d_{(G,w)}(\{s\})$ is even, there is no cut-edge[1] incident to s, and (G, w) is connected, then there exists a *local* connectivity preserving complete g-splitting-off at the vertex s from (G, w).

We compare and contrast complete h-splitting-off at a vertex from a hypergraph and complete g-splitting-off at a vertex from a graph. Complete h-splitting-off at a vertex s from a hypergraph enables the vertex s to exit the hypergraph by informing its incident hyperedges about how to merge and trim themselves in order to preserve degrees. In this sense, the definition of complete h-splitting-off at a vertex from a hypergraph serves the same role as complete g-splitting-off at a vertex from a graph. On the other hand, there are important differences between the two notions. Firstly, complete h-splitting-off at a vertex from a *graph* may not necessarily be a graph (owing to the creation of hyperedges of size at least 3) while it is an easy

[1]Equivalently, for every edge $e \in \delta_G(s)$ with $w(e) = 1$, the removal of that edge does not disconnect the graph.

exercise to show that complete g-splitting-off at a vertex from a *graph* will necessarily be a graph. Secondly, local/global connectivity preserving complete g-splitting-off at a vertex from a graph may not exist—consider splitting-off the center vertex of a star graph.

2.2 Existence and algorithms

In this section, we state our result showing that a local connectivity preserving complete h-splitting-off at a vertex from a hypergraph always exists and can be computed in strongly polynomial time.

Theorem 2.1 ([15]). *Given a hypergraph $(G = (V, E), w_G : E \to \mathbb{Z}_+)$ and a vertex $s \in V$, there exists a strongly polynomial-time algorithm to find a local connectivity preserving complete h-splitting-off at s from (G, w_G).*

A distinction between Theorem 2.1 and the graph splitting-off results of Lovász and Mader is that Theorem 2.1 shows the existence of a local connectivity preserving complete h-splitting-off at a vertex from a hypergraph *without any assumptions on the hypergraph* whereas Lovász's and Mader's results hold only under certain technical assumptions on the graph. In several applications of their results, additional arguments are needed to address cases where those technical assumptions do not hold. For this reason, we believe that Theorem 2.1 could be useful in simplifying the arguments involved in some of the applications of Lovász's and Mader's graph splitting-off results.

2.3 Applications

In this section, we present two applications of Theorem 2.1.

2.3.1 Steiner rooted connected orientations

Orienting *hypergraphs* is a fundamental area in graph theory and combinatorial optimization (see Frank's book [50]) with far reaching implications. For example, Woodall's conjecture can be reformulated as a hypergraph orientation problem [1]; moreover, hypergraph orientation results have recently been used in coding theory [2]. Király and Lau [74] showed that an approximate min-max relation holds for *rooted Steiner-connected orientations* in hypergraphs. To state their result, we need some terminology in hypergraph orientations.

Let $G = (V, E)$ be a hypergraph. An *orientation* $\overrightarrow{G} = (V, E, \texttt{head} : E \to V)$ of G is a directed hypergraph obtained by assigning a unique head vertex $\texttt{head}(e) \in e$ for each $e \in E$. A pair $(e, \texttt{head}(e))$ is denoted as a *hyperarc* with the head of the hyperarc being $\texttt{head}(e)$ and the tails of the hyperarc being $e \setminus \texttt{head}(e)$. Let $G = (V, E)$ be a hypergraph, $T \subseteq V$ be a set of terminals, $r \in T$ be a root vertex, and k be a positive integer. An orientation $\overrightarrow{G}$ of G is defined to be *Steiner rooted k-hyperarc-connected* if there exist k hyperarc-disjoint paths in $\overrightarrow{G}$ from t to r for every terminal $t \in T \setminus \{r\}$. Here, a path from t to r in a directed hypergraph is an alternating sequence of distinct vertices and hyperarcs $t = v_1, a_1, v_2, a_2, ..., a_{\ell-1}, v_\ell = r$ such that v_i is a tail of a_i and v_{i+1} is the head of a_i for every $i \in [\ell - 1]$. We say that a hypergraph G is *Steiner k-hyperedge-connected* if $\lambda_G(u, v) \geq k$ for every pair of distinct terminals $u, v \in T$. It is clear that if the hypergraph G has a Steiner rooted k-hyperarc-connected orientation, then G should be Steiner k-hyperedge-connected. However, the converse is not necessarily true. Király and Lau [74] showed that if the hypergraph is Steiner $2k$-hyperedge-connected, then it has a Steiner rooted k-hyperarc-connected orientation.

Theorem 2.2 (Király and Lau [74]). *Let $G = (V, E)$ be a hypergraph, $T \subseteq V$ be a subset of terminals, $r \in T$ be the root vertex, and k be a positive integer. If G is Steiner $2k$-hyperedge-connected, then it has a Steiner rooted k-hyperarc-connected orientation.*

Király and Lau's proof of Theorem 2.2 was based on careful uncrossing and contractions. In [15], we give an alternative proof of Theorem 2.2 using Theorem 2.1. Our proof of Theorem 2.2 reveals the source of the 2-factor gap in the approximate min-max relation of Király and Lau for MAX STEINER ROOTED-CONNECTED

Orientation Problem: it arises from the 2-factor gap between connectivity and *weak-partition-connectivity* of hypergraphs (see [15] for formal details). We note that an important special case of Theorem 2.2 is when the input hypergraph is a graph. In this case, the proof of the statement follows as a direct consequence of the Nash-Williams Strong orientation Theorem [94] which states that every undirected graph $G = (V, E)$ admits an orientation $\overrightarrow{G}$ such that $\lambda_{\overrightarrow{G}}(u, v) \geq \lfloor \lambda_G(u, v)/2 \rfloor$ for every distinct $u, v \in V$, where $\lambda_{\overrightarrow{G}}(u, v)$ is the maximum number of arc-disjoint directed paths from u to v in $\overrightarrow{G}$. Our proof strategy is thus unique since it proves an orientation result for *graphs* using tools developed for *hypergraphs*.

Remark 2.1. *Nash-Williams' proof of the strong orientation theorem [94] is a sophisticated inductive argument. Giving a simple and more insightful proof of the strong orientation theorem has been a central topic of interest in graph theory and combinatorial optimization (see [48]). Mader [87] gave a different proof of the strong orientation theorem using his local connectivity preserving splitting-off theorem, but his proof also involved sophisticated technical arguments. Frank [48] condensed the ideas of both Nash-Williams and Mader to present a proof of the strong orientation theorem using Mader's local connectivity preserving splitting-off, but it is still technically complicated. The technical complication in using Mader's local connectivity preserving splitting-off result arises from the assumptions that need to be satisfied by the vertex to be split-off. In contrast, our splitting-off result for hypergraphs (namely, Theorem 2.1) does not need any assumptions on the vertex to be split-off. In light of these considerations, our proof of Theorem 2.2 using Theorem 2.1 provides hope that Theorem 2.1 (or the ideas therein) could be used to give a conceptually simpler proof of Nash-Williams' strong orientation theorem.*

Our proof technique for Theorem 2.2 using Theorem 2.1 also leads to a simple alternate proof of Menger's theorem in *undirected* graphs and hypergraphs (edge-disjoint version). Moreover, Theorem 2.2 can be extended to weighted graphs/hypergraphs (by considering parallel copies of edges/hyperedges). This version of Theorem 2.2 can also be shown to admit strongly polynomial-time algorithms using our proof strategy as well as the proof strategy of Király and Lau. We avoid stating the weighted version in the interests of brevity.

2.3.2 Constructive characterizations

For the purposes of this application, we will use graphs and hypergraphs to refer to multi-graphs and multi-hypergraphs respectively. Let k be a positive integer. A graph $G = (V, E)$ is *k-edge-connected* if $d_G(X) \geq k$ for every non-empty proper subset $X \subsetneq V$. Constructive characterization of k-edge-connected graphs is a central problem in graph theory. It is well-known that a graph is 1-edge-connected if and only if it admits a spanning tree. Robbins' [101] showed that a graph is 2-edge-connected if and only if it admits an ear decomposition (see [50] for definition of ear decomposition). Generalizing Robbins' result, Lovász [84,86] gave a constructive characterization of k-edge-connected graphs for *even* k using his result on global connectivity preserving complete g-splitting-off at a vertex from a graph. Mader [87] gave a constructive characterization of k-edge-connected graphs for *odd* k using his result on local connectivity preserving complete g-splitting-off at a vertex from a graph. Motivated by these results, we present a constructive characterization of k-hyperedge-connected hypergraphs using our splitting-off result in Theorem 2.1. A hypergraph $G = (V, E)$ is defined to be *k-hyperedge-connected* if $d_G(X) \geq k$ for every non-empty proper subset $X \subsetneq V$.

Both Lovász's and Mader's constructive characterizations of k-edge-connected graphs are based on a pinching operation in graphs. Our constructive characterization of k-hyperedge-connected hypergraphs is also based on a pinching operation, but our pinching operation is defined for hypergraphs. We define this operation now (see Figure 1 for an example).

Definition 2.3 ([15]). *Let $G = (V, E)$ be a hypergraph and $p, k \in \mathbb{Z}_+$ be positive integers such that $p \leq k$. In* (k, p)-pinching *hyperedges of G, we obtain a new hypergraph by performing the following sequence of operations:*

1. *pick p distinct hyperedges $e_1, \ldots, e_p \in E$,*
2. *pick p positive integers $t_1, \ldots, t_p \in \mathbb{Z}_+$ such that $\sum_{i=1}^{p} t_i = k$,*

3. *for each $i \in [p]$, choose a partition of the hyperedge e_i into t_i non-empty parts $e_i = \uplus_{j\in[t_i]} f_i^j$,*

4. *remove the hyperedges $e_1, \ldots, e_p$ from the hypergraph G,*

5. *add a new vertex s and hyperedges $\{f_i^j \cup \{s\} : j \in [t_i], i \in [p]\}$ to the hypergraph G.*

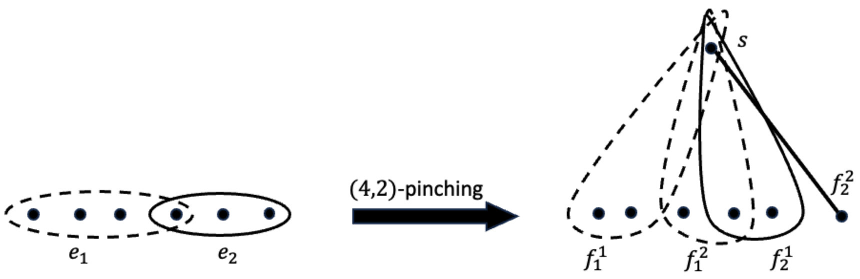

Figure 1: An example of a $(4, 2)$-pinching operation. Here, $t_1 = t_2 = 2$.

With this definition of pinching, we show the following constructive characterization of k-hyperedge-connected hypergraphs.

Theorem 2.3 ([15]). *Let $k \in \mathbb{Z}_+$ be a positive integer. A hypergraph $G = (V, E)$ is k-hyperedge-connected if and only if G can be obtained by starting from the single vertex hypergraph with no hyperedges and repeatedly applying one of the following two operations:*

1. *add a new hyperedge over a subset of vertices of the existing hypergraph, and*

2. *(k, p)-pinching hyperedges of the existing hypergraph for some positive integer $p \leq k$.*

Our proof of Theorem 2.3 is constructive: i.e., given a k-hyperedge-connected hypergraph G, our proof gives a polynomial-time algorithm to construct a sequence of hypergraphs $G_0, G_1, G_2, \ldots, G_t$, where G_0 is the single vertex hypergraph with no hyperedges, $G_t = G$ and for each $i \in [t]$, the hypergraph G_i is obtained from G_{i-1} by either adding a new hyperedge over a subset of vertices in G_{i-1} or by (k, p)-pinching hyperedges in G_{i-1} for some positive integer $p \leq k$.

2.4 Weak to Strong Cover of Symmetric Skew-Supermodular Functions

In this section, we present a more general statement that implies Theorem 2.1. We begin with the definitions needed for the more general statement.

Definition 2.4. *Let V be a finite set, $p : 2^V \to \mathbb{Z}$ be a set function, and $(H = (V, E), w : E \to \mathbb{Z}_+)$ be a hypergraph.*

1. *The set function p*
   (a) *is* symmetric *if $p(X) = p(V - X)$ for every $X \subseteq V$, and*
   (b) *is* skew-supermodular *if for every $X, Y \subseteq V$, at least one of the following inequalities hold:*
      i. $p(X) + p(Y) \leq p(X \cap Y) + p(X \cup Y)$.
      ii. $p(X) + p(Y) \leq p(X - Y) + p(Y - X)$.

2. *The coverage function $b_{(H,w)} : 2^V \to \mathbb{Z}_{\geq 0}$ is defined by $b_{(H,w)}(X) := \sum_{e\in B_H(X)} w(e)$ for every $X \subseteq V$, where $B_H(X) := \{e \in E : e \cap X \neq \emptyset\}$ for every $X \subseteq V$.*

3. *The hypergraph (H, w)* weakly covers *the function p if $b_{(H,w)}(X) \geq p(X)$ for every $X \subseteq V$.*

4. *The hypergraph (H, w)* strongly covers *the function p if $d_{(H,w)}(X) \geq p(X)$ for every $X \subseteq V$.*

Bernáth and Király [17] showed that a weak cover of a *symmetric skew-supermodular* function can be converted to a strong cover of the same function by repeated *merging of disjoint hyperedges.* We recall their definition of the merging operation, discuss their result, and its significance now.

Definition 2.5. *Let $(H = (V, E), w : E \to \mathbb{Z}_+)$ be a hypergraph. We use H_w to denote the unweighted multi-hypergraph over vertex set V containing $w(e)$ copies of every hyperedge $e \in E$. By* merging two disjoint hyperedges *of H_w, we refer to the operation of replacing them by their union in H_w. We will say that a hypergraph $(G = (V, E_G), c : E_G \to \mathbb{Z}_+)$ is obtained from (H, w) by* merging hyperedges *if the multi-hypergraph G_c is obtained from the multi-hypergraph H_w by repeatedly merging two disjoint hyperedges in the current hypergraph.*

Theorem 2.4 (Bernáth and Király [17])**.** *Let $(H = (V, E), w : E \to \mathbb{Z}_+)$ be a hypergraph and $p : 2^V \to \mathbb{Z}$ be a symmetric skew-supermodular function such that $b_{(H,w)}(X) \geq p(X)$ for every $X \subseteq V$. Then, there exists a hypergraph $\left(H^* = (V, E^*), w^* : E^* \to \mathbb{Z}_+\right)$ such that*

(1) *$d_{(H^*,w^*)}(X) \geq p(X)$ for every $X \subseteq V$ and*

(2) *the hypergraph (H^*, w^*) is obtained by merging hyperedges of the hypergraph (H, w).*

We observe that Theorem 2.4 can be used to prove the existential version of Theorem 2.1: namely, for every hypergraph $(G = (V, E), w_G : E \to \mathbb{Z}_+)$ and a vertex $s \in V$, *there exists* a local connectivity preserving complete h-splitting-off at s from (G, w_G). This can be shown by setting up the hypergraph (H, w) and the function p suitably based on (G, w_G) and using Theorem 2.4. We emphasize that this conclusion regarding hypergraph splitting-off from Bernáth and Király's result was not known before in the literature and is one of our contributions.

Remark 2.2. *We were also able to prove the existential version of Theorem 2.1 using* element-connectivity preserving reduction operations *[14] (see [33] for the definition of element-connectivity and the notion of element-connectivity preserving reduction operations). The alternate proof does not seem to be helpful for the purposes of a strongly polynomial time algorithm. In fact, it remains open to design a strongly polynomial-time algorithm to perform* complete *element-connectivity preserving reduction operations in the weighted setting [33].*

We recall that existence of a local connectivity preserving complete h-splitting-off at a vertex from a hypergraph does not immediately imply a polynomial-time algorithm However, the above-mentioned proof of existence of a local-connectivity preserving complete splitting-off at an arbitrary vertex from a hypergraph (i.e., existential version of Theorem 2.1) via Theorem 2.4 suggests a natural approach towards designing a strongly polynomial time algorithm to find a local-connectivity preserving complete splitting-off at a given vertex from a given hypergraph: it suffices to prove a constructive version of Theorem 2.4 via a strongly polynomial-time algorithm. In particular, we need to show Theorem 2.4 with the extra conclusion that the number of additional hyperedges in H^* is polynomial in the number of hyperedges and vertices in H.

Bernáth and Király proved Theorem 2.4 in the context of a reduction between certain hypergraph connectivity augmentation problems. For that reduction, the existential version of Theorem 2.4 is sufficient. However, for the purposes of our application to hypergraph splitting-off, we need an algorithmic version of Theorem 2.4. Bernáth and Király's proof of Theorem 2.4 is in fact algorithmic, but the run-time of the associated algorithm is not necessarily polynomial. Their proof implies that the number of additional hyperedges in the hypergraph (H^*, c^*) returned by their algorithm is at most $\sum_{e \in E} w(e)$ (i.e., $|E^*| - |E| \leq \sum_{e \in E} w(e)$) and the run-time of the algorithm is $O(\sum_{e \in E}(|e| + w(e))$. In particular, their run-time is polynomial only if the input weights are given in unary. In fact, in [15], we given example of an exponential-sized which could arise as in the execution of their algorithm.

We address both the structural and the algorithmic issues mentioned above by proving a stronger algorithmic version of Theorem 2.4. In order to phrase an algorithmic version of Theorem 2.4, we need suitable access to the function p. Bernáth and Király [17] suggested access to a certain function maximization oracle associated with the function p that we describe below.

Definition 2.6. *Let $p : 2^V \to \mathbb{Z}$ be a set function.* p-`max-sc-Oracle` $\left((G_0, c_0), S_0, T_0\right)$ *takes as input a hypergraph $(G_0 = (V, E_0), c_0 : E_0 \to \mathbb{Z}_+)$ and disjoint sets $S_0, T_0 \subseteq V$, and returns a tuple $(Z, p(Z))$, where Z is an optimum solution to the following problem:*

$$\max\left\{p(Z) - d_{(G_0,c_0)}(Z) : S_0 \subseteq Z \subseteq V - T_0\right\}. \qquad (p\texttt{-max-sc-Oracle})$$

For the purposes of our application (namely local connectivity preserving complete h-splitting-off at a vertex from a hypergraph), the above-mentioned function maximization oracle can be implemented to run in strongly polynomial time. Using the above mentioned oracle, we prove the following algorithmic version of Theorem 2.4.

Theorem 2.5 ([15]). *Let $(H = (V, E), w : E \to \mathbb{Z}_+)$ be a hypergraph and $p : 2^V \to \mathbb{Z}$ be a symmetric skew-supermodular function such that $b_{(H,w)}(X) \geq p(X)$ for every $X \subseteq V$. Then, there exists a hypergraph $\left(H^* = (V, E^*), w^* : E^* \to \mathbb{Z}_+\right)$ such that*

(1) *$d_{(H^*,w^*)}(X) \geq p(X)$ for every $X \subseteq V$,*

(2) *the hypergraph (H^*, w^*) is obtained by merging hyperedges of the hypergraph (H, w), and*

(3) *$|E^*| - |E| = O(|V|)$.*

Furthermore, given a hypergraph $(H = (V, E), w : E \to \mathbb{Z}_+)$ and access to p-`max-sc-Oracle` of a symmetric skew-supermodular function $p : 2^V \to \mathbb{Z}$ where $b_{(H,w)}(X) \geq p(X)$ for every $X \subseteq V$, there exists an algorithm that runs in $O(|V|^3(|V| + |E|)^2)$ time using $O(|V|^3(|V| + |E|))$ queries to p-`max-sc-Oracle` and returns a hypergraph $\left(H^* = (V, E^*), w^* : E^* \to \mathbb{Z}_+\right)$ satisfying the above three properties. The run-time includes the time to construct the hypergraphs used as input to the queries to p-`max-sc-Oracle`. Moreover, for each query to p-`max-sc-Oracle`, the hypergraph (G_0, c_0) used as input to the query has $O(|V|)$ vertices and $O(|V| + |E|)$ hyperedges.

Theorem 2.5 is a strengthening of Theorem 2.4 in two ways. Firstly, our theorem shows the existence of a hypergraph that not only satisfies properties (1) and (2), but also satisfies property (3) – i.e., the number of additional hyperedges in the returned hypergraph is *linear* in the size of the vertex set. Secondly, our Theorem 2.5 shows the existence of a strongly polynomial-time algorithm that returns a hypergraph satisfying the three properties. Our main contribution is modifying Bernáth and Király's algorithm and analyzing the modified algorithm to bound the number of additional hyperedges and the run-time. We mention that property (3) cannot be tightened to guarantee that $|E^* - E| = O(|V|)$. In [15] we show an example where $|E^* - E| = \Omega(|V|^2)$.

Theorem 2.5 immediately leads to a proof of Theorem 2.1. Instead of using Theorem 2.5 as a black-box, if we delve into the proof of it in the context of the proof of Theorem 2.1, we obtain the following theorem:

Theorem 2.6 ([15]). *Let $(G = (V, E), w : E \to \mathbb{Z}_+)$ be a hypergraph and $s \in V$. Then, there exists a hypergraph $(G' = (V, E'), w' : E' \to \mathbb{Z}_+)$ obtained by applying a h-splitting-off operation at s from (G, w) such that $\lambda_{(G',w')}(u, v) = \lambda_{(G,w)}(u, v)$ for every distinct $u, v \in V \setminus \{s\}$.*

Theorem 2.6 closely resembles the *existential edge splitting-off* results of Lovász [84, 86] and Mader [87] for graphs. Lovász's and Mader's existential edge splitting-off results for graphs are important since they have been used to simplify the proofs of fundamental results in graph theory—e.g., Nash-Williams' Strong Orientation Theorem. On the other hand, Theorem 2.6 does not immediately imply a strongly polynomial-time algorithm for finding a local connectivity preserving complete h-splitting off at a vertex from a given weighted hypergraph. So, Theorem 2.1 may be useful in algorithmic contexts while Theorem 2.6 may be useful in graph-theoretical contexts.

2.5 Open Problems

In this section, we list some open problems that arise as a result of our work.

1. Is there a fast algorithm to compute a local connectivity preserving complete h-splitting off? The crude runtime of our algorithm for Theorem 2.1 is $O(|V|^6(|V|+|E|)^3$. It is likely that recent developments in fast Gomory-Hu tree construction [64,81] can help improve the runtime by leveraging Gomory-Hu trees in a manner similar to that used by Lau and Yung [79] for fast local connectivity-preserving g-splitting off in graphs. We note that improving the runtime of local-connectivity preserving g-splitting off in graphs is a long-standing open problem.

2. Theorem 2.1 focused on local connectivity-preserving complete *h-splitting-off* at a vertex from a hypergraph. We mentioned the star graph as an example showing that local/global connectivity-preserving complete *g-splitting-off* at a vertex from a hypergraph may not exist. Are there *sufficient conditions* to guarantee the existence of such an operation? We recall that Lovász's [84,86] and Mader's [87] results give sufficient conditions to guarantee local and global connectivity-preserving complete g-splitting-off at a vertex from a *graph*.

3. In graphs and hypergraphs, it is known that there exists a global-connectivity preserving g-splitting-off of *edges* that respects *partition* constraints, i.e. the edges added during the split-off operations cannot have both endpoints belonging to the same part of a specified partition [5]. This has also been extended to the setting of covering symmetric crossing supermodular functions (using graph edges) [16]. Are there extensions of these results for hypergraphs? In particular, does there exist a global-connectivity preserving complete h-splitting off that respects *partition* constraints? A notion of respecting partition constraints here is that all hyperedges that are merged together via h-merge operations must belong to separate parts.

4. Is there an appropriate definition of complete splitting-off that would preserve the rank and the local-connectivities of the hypergraph?

5. Is there an appropriate definition of complete splitting-off in directed hypergraphs that would preserve the local-connectivities of the hypergraph?

6. Goemans [56] showed that there exists a complete splitting-off that preserves all 4/3-*approximate* min-cuts of a graph. Does there exist a complete h-splitting-off that similarly preserves *approximate* min-cuts in hypergraphs?

7. Benczur and Goemans [13] showed that there exists a polygonal representation of (approximate) minimum cuts of a graph. How does the structure of the polygonal representation change during splitting-off operations? Can the polygon be maintained efficiently? Moreover, can this be used to speed up the computation of local-connectivity preserving complete splitting-off?

8. Recently, splitting-off in graphs has been used to design tools that have been used in designing a polynomial time randomized approximation scheme for congruency-constrained minimum cut [95]. Can local-connectivity preserving h-splitting-off be used to develop analogous tools for approximating the congruency-constrained minimum cut in hypergraphs?

9. We reiterate the question posed in Remark 2.2. The notion of *element-connectivity* in graphs closely related to hypergraph connectivity. There exists a notion of *element connectivity preserving reduction operations* which is analogous to connectivity-preserving splitting off in graphs and hypergraphs [33,66]. Is it possible to perform *complete* element-connectivity preserving reduction operations in the *weighted* setting in strongly polynomial time?

3 Hypergraph connectivity augmentation

In this section, we present our results on hypergraph connectivity augmentation which appeared at ESA 2024 [22].

3.1 Degree-Specified Hypergraph Connectivity Augmentation

We recall the variant of DS-Graph-CA-using-E for hypergraphs which will be the focus of this section.

Definition 3.1 (DS-Hypergraph-CA-using-H)**.** Degree-specified Hypergraph Connectivity Augmentation using Hyperedges problem *is defined as follows:*

Input:	*A hypergraph* $(G = (V, E_G, c_G : E_G \to \mathbb{Z}_+)$, *target connectivity function* $r : \binom{V}{2} \to \mathbb{Z}_{\geq 0}$, *and degree requirement function* $m : V \to \mathbb{Z}_{\geq 0}$.
Goal:	*Verify if there exists a hypergraph* $(H = (V, E_H), w_H : E_H \to \mathbb{Z}_+)$ *such that* $b_{(H,w_H)}(u) = m(u)$ *for every* $u \in V$, $\lambda_{(G+H,c_G+w_H)}(u,v) \geq r(u,v)$ *for every distinct* $u, v \in V$, *and if so, then find such a hypergraph.*

Our main result below strengthens both the structural and algorithmic results of Szigeti [106] and shows that feasible instances admit solutions with $O(|V|)$ hyperedges and can be computed in strongly polynomial time algorithm.

Theorem 3.1 ([22])**.** *There exists an algorithm to solve DS-Hypergraph-CA-using-H that runs in time* $O(n^7(n+m)^2)$*, where* n *is the number of vertices and* m *is the number of hypergedges in the input hypergraph. Moreover, if the instance is feasible, then the algorithm returns a solution hypergraph that contains at most* $4n$ *hyperedges.*

Next, we consider a variant of DS-Hypergraph-CA-using-H which requires solution hypergraphs to be near-uniform—we recall that a hypergraph is *uniform* if all hyperedges have the same size; a hypergraph is *near-uniform* if every pair of hyperedges differ in size by at most one. Our next result shows that feasible instances of DS-Hypergraph-CA-using-H admit solution hypergraphs with $O(|V|)$ near-uniform hyperedges; moreover, these solution hypergraphs can be computed in strongly polynomial time.

Theorem 3.2 ([22])**.** *There exists a strongly polynomial time algorithm to solve DS-Hypergraph-CA-using-H. Moreover, if the instance is feasible, then the algorithm returns a solution hypergraph that is near-uniform and contains* $O(n)$ *hyperedges, where* n *is the number of vertices in the input hypergraph.*

Next, we consider a variant of hypergraph connectivity augmentation where the goal is to simultaneously augment two input hypergraphs to achieve certain target connectivities using the same degree-specified hypergraph.

Definition 3.2 (DS-Simul-Hypergraph-CA-using-H)**.** *The* Degree-specified Simultaneous Hypergraph Connectivity Augmentation Hyperedges *is defined as follows:*

Given:	*Hypergraphs* $(G_i = (V, E_i), c_i : E_i \to \mathbb{Z}_+)$ *for* $i \in \{1, 2\}$, *target connectivity functions* $r_i : \binom{V}{2} \to \mathbb{Z}_{\geq 0}$ *for* $i \in \{1, 2\}$ *such that* $\max\{r_1(u,v) - \lambda_{(G_1,c_1)}(u,v) : u, v \in V\} = \max\{r_2(u,v) - \lambda_{(G_2,c_2)}(u,v) : u, v \in V\}$, *and degree requirement function* $m : V \to \mathbb{Z}_{\geq 0}$.
Goal:	*Verify if there exists a hypergraph* $(H = (V, E_H), w_H : E_H \to \mathbb{Z}_+)$ *such that* $b_{(H,w_H)}(u) = m(u)$ *for every* $u \in V$ *and* $\lambda_{(G_i+H,c_i+w_H)}(u,v) \geq r_i(u,v)$ *for every distinct* $u, v \in V$ *and* $i \in \{1, 2\}$, *and if so, then find such a hypergraph.*

Bernáth and Király [17] proposed the above problem and showed that if the assumption $\max\{r_1(u,v) - \lambda_{(G_1,c_1)}(u,v) : u,v \in V\} = \max\{r_2(u,v) - \lambda_{(G_2,c_2)}(u,v) : u,v \in V\}$ does not hold, then the problem is NP-complete. So, we include this assumption in the definition of the problem statement. Our result below extends both the structural and algorithmic results of Bernáth and Király for DS-Simul-Hypergraph-CA-using-H.

Theorem 3.3 ([22]). *There exists a strongly polynomial time algorithm to solve DS-Simul-Hypergraph-CA-using-H. Moreover, if the instance is feasible, then the algorithm returns a solution hypergraph that is near-uniform and contains $O(n^2)$ hyperedges, where n is the number of vertices in the input hypergraph.*

Our algorithm for Theorems 3.2 and 3.3 are LP-based. The associated LPs can be solved in strongly polynomial time. However, the LP-solving time is large, so we refrain from stating the run-times explicitly. In contrast, our algorithm for Theorem 3.1 is combinatorial and hence, we are able to provide an explicit bound on the run-time. We refer the reader to Table 1 for a list of graph/hypergraph connectivity augmentation problems using edges/hyperedges, previously known results, and our results.

Problem	**Complexity Status**
DS-Graph-CA-using-E	Strong Poly [47]
DS-Hypergraph-CA-using-E	NP-comp [41,88]
DS-Hypergraph-CA-using-H	Psuedo Poly [106] $O(n^7(n+m)^2)$ time (Thm 3.1)
DS-Hypergraph-CA-using-near-uniform-H	Pseudo Poly [17] Strong Poly (Thm 3.2)
DS-Simul-Hypergraph-CA-using-H	Pseudo Poly [17] Strong Poly (Thm 3.3)
DS-Simul-Hypergraph-CA-using-near-uniform-H	Pseudo Poly [17] Strong Poly (Thm 3.3)

Table 1: Complexity of Graph and Hypergraph Connectivity Augmentation Problems using Edges and Hyperedges. Problems having "Near-Uniform" in their title are similar to the corresponding problems without "Near-Uniform" in their title but have the additional requirement that the returned solution hypergraph be *near-uniform*. Here, n and m denote the number of vertices and hyperedges respectively in the input hypergraph.

3.2 Degree-Specified Skew-Supermodular Cover Problems

In this section we present extensions of the results from the previous section to a general function cover problem. We first recall certain definitions needed to describe the general function cover problems.

Definition 3.3. *Let V be a finite set, $(H = (V,E), w : E \to \mathbb{Z}_+)$ be a hypergraph, and $p : 2^V \to \mathbb{Z}$ be a set function.*

1. *The hypergraph (H,w)* weakly covers *the function p if $b_{(H,w)}(X) \ge p(X)$ for every $X \subseteq V$.*
2. *The hypergraph (H,w)* strongly covers *the function p if $d_{(H,w)}(X) \ge p(X)$ for every $X \subseteq V$.*

We will be interested in the problem of finding a degree-specified hypergraph that strongly/weakly covers a given function p. In all our applications (including DS-Hypergraph-CA-using-H), the function p of interest will be skew-supermodular and/or symmetric.

Definition 3.4. *Let $p : 2^V \to Z$ be a set function. We will denote the maximum function value of p by K_p, i.e., $K_p := \max\{p(X) : X \subseteq V\}$. The set function p*

1. *is* symmetric *if $p(X) = p(V - X)$ for every $X \subseteq V$, and*

2. *is* skew-supermodular *if for every $X, Y \subseteq V$, at least one of the following inequalities hold:*
   (a) $p(X) + p(Y) \leq p(X \cap Y) + p(X \cup Y)$. *If this inequality holds, then we say that p is* locally supermodular *at X, Y.*
   (b) $p(X) + p(Y) \leq p(X - Y) + p(Y - X)$. *If this inequality holds, then we say that p is* locally negamodular *at X, Y.*

We will assume access to the skew-supermodular function p via the following oracle.

Definition 3.5. *Let $p : 2^V \to \mathbb{Z}$ be a set function. p-`max-sc-Oracle` $\left((G_0, c_0), S_0, T_0, y_0\right)$ takes as input a hypergraph $(G_0 = (V, E_0), c_0 : E_0 \to \mathbb{Z}_+)$, disjoint sets $S_0, T_0 \subseteq V$, and a vector $y_0 \in \mathbb{R}^V$; the oracle returns a tuple $(Z, p(Z))$, where Z is an optimum solution to the following problem:*

$$\max\left\{p(Z) - d_{(G_0,c_0)}(Z) + y_0(Z) : S_0 \subseteq Z \subseteq V - T_0\right\}. \qquad (p\text{-}\texttt{max-sc-Oracle})$$

We note that p-`max-sc-Oracle` is strictly stronger than the function evaluation oracle[2]: function evaluation oracle can be implemented using one query to p-`max-sc-Oracle` while it is impossible to maximize a skew-supermodular function using polynomial number of queries to its function evaluation oracle [61]. However, we will see later that p-`max-sc-Oracle` can indeed be implemented in strongly polynomial time for the functions p of interest to our applications. In our algorithmic results, we will ensure that the size of hypergraphs (G_0, c_0) used as inputs to p-`max-sc-Oracle` are polynomial in the input size (in particular, the number of hyperedges in these hypergraphs will be polynomial in the size of the ground set V). We now describe the general function cover problems that will be of interest to this work.

Strong Cover Problem. In all our applications, we will be interested in obtaining a degree-specified strong cover of a function in strongly polynomial time.

Definition 3.6 (DS-Sym-Skew-SupMod-StrongCover-using-H)**.** Degree-specified symmetric skew-supermodular strong cover using hyperedges problem *is defined as follows:*

Input:	*A degree requirement function $m : V \to \mathbb{Z}_{\geq 0}$ and a $\underline{\text{symmeric}}$ skew-supermodular function $p : 2^V \to \mathbb{Z}$ via p-`max-sc-Oracle`.*
Goal:	*Verify if there exists a hypergraph $(H = (V, E), w : E \to \mathbb{Z}_+)$ such that $b_{(H,w)}(u) = m(u)$ for every $u \in V$ and (H, w) $\underline{\text{strongly}}$ covers the function p, and if so, then find such a hypergraph.*

DS-Sym-Skew-SupMod-StrongCover-using-H was introduced by Bernáth and Király [17] as a generalization of DS-Hypergraph-CA-using-H. They showed that it is impossible to solve DS-Sym-Skew-SupMod-StrongCover-using-H using polynomial number of queries to the function evaluation oracle. They suggested access to p-`max-sc-Oracle` and we work in the same function access model as Bernáth and Király. We note that it is not immediately clear if feasible instances of DS-Sym-Skew-SupMod-StrongCover-using-H admit solution hypergraphs with polynomial number of hyperedges , so membership of the problem in NP is not obvious.

Weak Cover Problem. Although our applications will be concerned with degree-specified *strong* cover, our techniques will be concerned with degree-specified *weak* cover problems.

Definition 3.7 (DS-Skew-SupMod-WeakCover-using-H)**.** Degree-specified skew-supermodular weak cover using hyperedges problem *is defined as follows:*

[2]For a function $p : 2^V \to \mathbb{Z}$, the function evaluation oracle takes a subset $X \subseteq V$ as input and returns $p(X)$.

Input:	*A degree requirement function* $m : V \to \mathbb{Z}_{\geq 0}$ *and* *a skew-supermodular function* $p : 2^V \to \mathbb{Z}$ *via* p-`max-sc-Oracle`.
Goal:	*Verify if there exists a hypergraph* $(H = (V, E), w : E \to \mathbb{Z}_+)$ *such that* $\sum_{e \in E} w(e) = K_p$, $b_{(H,w)}(u) = m(u)$ *for every* $u \in V$, *and* (H, w) *weakly covers the function* p, *and if so, then find such a hypergraph.*

We note that if a hypergraph $(H = (V, E), w : E \to \mathbb{Z}_+)$ strongly covers a function $p : 2^V \to \mathbb{Z}$, then it also weakly covers the function p; the converse statement is false[3]. However, imposing the constraint $\sum_{e \in E} w(e) = K_p$ implies the converse—we elaborate on this now. We note that the requirement $\sum_{e \in E} w(e) = K_p$ is present in DS-Skew-SupMod-WeakCover-using-H but not in DS-Sym-Skew-SupMod-StrongCover-using-H. Firstly, if we drop this constraint from the definition of DS-Skew-SupMod-WeakCover-using-H, then feasible instances of the resulting problem admit trivial solutions[4]. Thus, imposing this constraint makes the problem non-trivial. Secondly, Szigeti [106] showed that if an instance of DS-Skew-SupMod-WeakCover-using-H is feasible, then it admits a solution hypergraph $(H = (V, E), w : E \to \mathbb{Z}_+)$ satisfying the constraint $\sum_{e \in E} w(e) = K_p$. Moreover, Bernáth and Király [17] observed that if a hypergraph $(H = (V, E), w : E \to \mathbb{Z}_+)$ with $\sum_{e \in E} w(e) = K_p$ weakly covers a symmetric skew-supermodular function p, then (H, w) also strongly covers p. Thus, the converse of the previously mentioned relationship between weak and strong covers is in fact true after imposing the constraint. The observations of Szigeti [106] and Bernath and Kiraly [17] together imply that in order to solve DS-Sym-Skew-SupMod-StrongCover-using-H, it suffices to solve the DS-Skew-SupMod-WeakCover-using-H problem for the same function p. Our result below shows that feasible instances of DS-Skew-SupMod-WeakCover-using-H admit a solution hypergraphs with $O(|V|)$ distinct hyperedges; moreover, these solution hypergraphs can be computed in strongly polynomial time. Thus, our result also shows implies a strongly polynomial-time algorithm for DS-Sym-Skew-SupMod-StrongCover-using-H.

Theorem 3.4 ([22]). *There exists an algorithm to solve DS-Skew-SupMod-WeakCover-using-H that runs in time* $O(|V|^5)$ *using* $O(|V|^4)$ *queries to* p-`max-sc-Oracle`, *where* V *is the ground set of the input instance. Moreover, if the instance is feasible, then the algorithm returns a solution hypergraph that contains at most* $4|V|$ *hyperedges. For each query to* p-`max-sc-Oracle` *made by the algorithm, the hypergraph* (G_0, c_0) *used as input to the query has* $O(|V|)$ *vertices and* $O(|V|)$ *hyperedges.*

Near-Uniform and Simultaneous Covers. Bernáth and Király [17] strengthened Szigeti's result [106] via an LP-based approach. They showed that if an instance of DS-Skew-SupMod-WeakCover-using-H is feasible, then it admits a solution hypergraph that is near-uniform. Their approach is also algorithmic, but the run-time of their algorithm is only pseudo-polynomial (again, owing to the dependence on K_p which may not be polynomial in $|V|$). Our next result shows that feasible instances of DS-Skew-SupMod-WeakCover-using-H admit solutions with *linear* number of distinct, *near-uniform* hyperedges. Moreover, such solution hypergraphs can be constructed in strongly polynomial time.

Theorem 3.5 ([22]). *There exists an algorithm to solve DS-Skew-SupMod-WeakCover-using-H that runs in* $\text{poly}(|V|)$ *time using* $\text{poly}(|V|)$ *queries to* p-`max-sc-Oracle`, *where* V *is the ground set of the input instance. Moreover, if the instance is feasible, then the algorithm returns a solution hypergraph that is near-uniform and contains* $O(|V|)$ *hyperedges. For each query to* p-`max-sc-Oracle` *made by the algorithm, the hypergraph* (G_0, c_0) *used as input to the query has* $O(|V|)$ *vertices and* $O(|V|)$ *hyperedges.*

Bernáth and Király's [17] LP-based approach also helped in addressing the problem of *simultaneous* weak covers—namely, DS-Simul-Skew-SupMod-WeakCover-using-H—where the goal is to compute a hypergraph that satisfies degree requirements and is simultaneously a weak cover for two given input functions.

[3]For example, consider the function $p : 2^V \to \mathbb{Z}$ defined by $p(X) := 1$ for every non-empty proper subset $X \subsetneq V$ and $p(\emptyset) := p(V) := 0$, and the hypergraph $(H = (V, E := \{\{u\} : u \in V\}), w : E \to \{1\})$.

[4]Consider the hypergraph $(H = (V, E), w : E \to \mathbb{Z}_+)$, where $E := \{\{u\} : u \in V,\ m(u) \geq 1\}$ with $w(\{u\}) := m(u)$ for every $\{u\} \in E$.

Problem	**Complexity Status**
DS-Skew-SupMod-WeakCover-using-H	Pseudo Poly [106] Strong Poly (Thm 3.4)
DS-Sym-Skew-SupMod-StrongCover-using-H	Pseudo Poly [106] Strong Poly*
DS-Skew-SupMod-WeakCover-using-near-uniform-H	Pseudo Poly [17] Strong Poly (Thm 3.5)
DS-Sym-Skew-SupMod-StrongCover-using-near-uniform-H	Pseudo Poly [17] Strong Poly*
DS-Simul-Skew-SupMod-WeakCover-using-near-uniform-H	Pseudo Poly [17] Strong Poly (Thm 3.6)
DS-Simul-Sym-Skew-SupMod-StrongCover-using-near-uniform-H	Pseudo Poly [17] Strong Poly*

Table 2: Complexity of degree-specified skew-supermodular cover using hyperedges problems. Problems having "Near-Uniform" in their title are similar to the corresponding problems without "Near-Uniform" in their title but have the additional requirement that the returned solution hypergraph be *near-uniform*. Results marked with an asterisk are not included in this proposal for brevity, but can be found in the complete version of the ICALP'24 paper [22].

They gave a complete characterization for the existence of a feasible solution to DS-Simul-Skew-SupMod-WeakCover-using-H. Moreover, they showed that if an instance is feasible, then it admits a solution hypergraph that is near-uniform. Their proof is algorithmic, but the run-time of their algorithm is only pseudo-polynomial (again, owing to the dependence on K_p). Our next result shows that feasible instances of DS-Simul-Skew-SupMod-WeakCover-using-H admit solutions with at most *quadratic* number of distinct, *near-uniform* hyperedges. Moreover, such solution hypergraphs can be constructed in strongly polynomial time.

Theorem 3.6 ([22]). *There exists an algorithm to solve DS-Simul-Skew-SupMod-WeakCover-using-H that runs in* $\text{poly}(|V|)$ *time using* $\text{poly}(|V|)$ *queries to* q-`max-sc-Oracle` *and* r-`max-sc-Oracle`*, where* V *is the ground set of the input instance. Moreover, if the instance is feasible, then the algorithm returns a solution hypergraph that is near-uniform and contains* $O(|V|^2)$ *hyperedges. For each query to* q-`max-sc-Oracle` *and* r-`max-sc-Oracle` *made by the algorithm, the hypergraph* (G_0, c_0) *used as input to the query has* $O(|V|)$ *vertices and* $O(|V|^2)$ *hyperedges.*

Our algorithm for Theorem 3.4 is combinatorial and hence, we are able to provide an explicit bound on the run-time. In contrast, our algorithms for Theorems 3.5 and 3.6 are LP-based. These LPs are solvable in strongly polynomial time, but their run-time is large, so we refrain from stating the run-times explicitly. We refer to Table 2 for a list of degree-specified skew-supermodular cover using hyperedges problems, previously known results, and our results. The results for strong cover problems follow from our results for weak cover problems.

3.3 Applications

Theorems 3.2 and 3.3 can be seen as corollaries of Theorems 3.5 and 3.6 respectively by reducing the augmentation problems to appropriate function cover problems. In this section, we provide two additional applications of Theorem 3.4. We note that the corresponding variants involving simultaneous augmentation and near-uniform hyperedges can also be defined and solved in strongly polynomial time as applications of Theorems 3.5 and 3.6. Moreover, the appropriate optimization variants of the applications can also be solved in strongly polynomial time by reduction to the degree-specified version of the problems and applying our results.

3.3.1 Degree-specified hypergraph node-to-area connectivity augmentation using hyperedges

The graph node-to-area connectivity augmentation using edges problem was solved by Ishii and Hagiwara [67]. Bernáth and Király [17] proposed a hypergraphic variant of this problem that we describe now.

Definition 3.8 (DS-HyperGraph-Node-to-Area-CA-using-H)**.** *The* degree-specified hypergraph node-to-area connectivity augmentation using hyperedges *problem is defined as follows:*

Input:	*Degree requirement function* $m : V \to \mathbb{Z}_{\geq 0}$, *hypergraph* $(G = (V, E), c : E \to \mathbb{Z}_+)$, *a collection* $\mathcal{W} \subseteq 2^V$ *of subsets of* V, *and target connectivity function* $r : \mathcal{W} \to \mathbb{Z}_+$.
Goal:	*Verify if there exists a hypergraph* $(H = (V, E_H), w : E_H \to \mathbb{Z}_+)$ *such that* $\lambda_{(G+H,c+w)}(u, W) \geq r(W)$ *for every* $W \in \mathcal{W}$ *and* $u \in V - W$, *and* $b_{(H,w)}(u) = m(u)$ *for every* $u \in V$, *and if so, then find such a hypergraph.*

Bernáth and Király [17] showed that feasible instances of DS-HyperGraph-Node-to-Area-CA-using-H admit a near-uniform hypergraph as a feasible solution and such a solution can be found in pseudo-polynomial time. Our next result implies a strongly polynomial-time algorithm for this problem.

Theorem 3.7. *There exists a strongly polynomial time algorithm to solve DS-HyperGraph-Node-to-Area-CA-using-H. Moreover, if the instance is feasible, then the algorithm returns a solution hypergraph that is near-uniform and contains $O(n)$ hyperedges, where n is the number of vertices in the input hypergraph.*

3.3.2 Degree-constrained mixed-hypergraph connectivity augmentation using hyperedges.

Preliminaries. The notion of mixed-hypergraph generalizes undirected graphs/hypergraphs and directed graphs/hypergraphs (e.g., see [50]). A mixed-hypergraph $(G = (V, A), c : A \to \mathbb{Z}_+)$ consists of a finite set V of vertices and a set A of hyperarcs with weights $c(e)$ for every $e \in A$, where each hyperarc $e \in A$ is specified by an ordered tuple $(\text{tails}(e), \text{heads}(e), \text{heads-tails}(e))$ satisfying $\text{tails}(e), \text{heads}(e), \text{heads-tails(e)} \subseteq V$ with $\text{tails}(e) \cap \text{heads}(e) = \emptyset$, $\text{tails}(e) \cap \text{heads-tails}(e) = \emptyset$, $\text{heads}(e) \cap \text{heads-tails}(e) = \emptyset$, $\text{tails}(e) \cup \text{heads-tails}(e) \neq \emptyset$ and $\text{heads}(e) \cup \text{heads-tails}(e) \neq \emptyset$. For an undirected hypergraph $(H = (V, E_H), w_H : E_H \to \mathbb{Z}_+)$, we define an associated mixed-hypergraph $(\overrightarrow{H} = (V, \overrightarrow{E_H}), w_{\overrightarrow{H}} : \overrightarrow{E_H} \to \mathbb{Z}_+)$ where for every hyperedge $e_H \in E_H$, we introduce a hyperarc $e_{\overrightarrow{H}} := (\text{tails}(e_{\overrightarrow{H}}) = \emptyset, \text{heads}(e_{\overrightarrow{H}}) = \emptyset, \text{heads-tails}(e_{\overrightarrow{H}}) = e_H)$ into $\overrightarrow{E_H}$ with weight $w_{\overrightarrow{H}}(e_{\overrightarrow{H}}) = w_H(e_H)$. For a set $X \subseteq V$, we define $\delta^{in}_G(X) := \{e \in A : (\text{heads}(e) \cup \text{heads-tails}(e)) \cap A \neq \emptyset, (\text{tails}(e) \cup \text{heads-tails}(e)) \cap (V \setminus A) \neq \emptyset\}$ and the function $d^{in}_{(G,w)} : 2^V \to \mathbb{Z}_{\geq 0}$ defined by $d^{in}_{(G,w)}(X) := \sum_{e \in \delta^{in}_G(X)} w(e)$ for every $X \subseteq V$. For distinct vertices $u, v \in V$, we define $\lambda_{(G,w)}(u, v) := \min\{d^{in}_{(G,w)}(X) : v \in X \subseteq V - \{u\}\}$. We note that $\lambda_{(G,w)}(u, v)$ is not necessarily equal to $\lambda_{(G,w)}(v, u)$ for distinct vertices $u, v \in V$. Let $r \in V$ be a designated root vertex. If $\lambda_{(G,w)}(v, r) \geq k$ and $\lambda_{(G,w)}(r, v) \geq \ell$ for every $v \in V \setminus \{r\}$, then the mixed-hypergraph is said to be (k, ℓ)-rooted-arc-connected.

Definition 3.9 (DC-Mixed-HyperGraph-Global-CA-using-H)**.** *The* degree-constrained mixed-hypergraph global connectivity augmentation using hyperedges *problem is defined as follows:*

Input:	*Degree constraint function* $m : V \to \mathbb{Z}_{\geq 0}$, *mixed-hypergraph* $(G = (V, A), c : A \to \mathbb{Z}_+)$, *root vertex* $r \in V$, *and integers* $k, \ell \in \mathbb{Z}_+$.
Goal:	*Verify if there exists a undirected hypergraph* $(H = (V, E_H), w : E_H \to \mathbb{Z}_+)$ *such that* $b_{(H,w)}(u) \leq m(u)$ *for every* $u \in V$, *and the mixed-hypergraph* $(G + \overrightarrow{H}, c + w_{\overrightarrow{H}})$ *is* (k, ℓ)*-rooted-arc-connected, and if so, then find such a hypergraph.*

Bernáth and Király [17] introduced this problem and showed a complete characterization for the existence of a feasible solution. They also showed that for feasible instances, there exists a feasible solution that is also near-uniform. Our next result implies a strongly polynomial-time algorithm for this problem.

Theorem 3.8. *There exists a strongly polynomial time algorithm to solve* DC-Mixed-HyperGraph-Global-CA-using-H. *Moreover if the instance is feasible, then the algorithm returns a solution hypergraph that is near-uniform and contains* $O(n)$ *hyperedges, where* n *is the number of vertices in the input hypergraph.*

3.4 Open Problems

In this section, we list some open problems that arise as a result of our work.

1. We mentioned that the past few years have seen exciting developments for DS-Graph-CA-using-E culminating in near-linear time algorithms for the problem [24, 25, 26]. Is it possible to get analogous results for hypergraphs? In particular, can we solve DS-Hypergraph-CA-using-H in near-linear time?

2. We define the size of a hypergraph to be the sum of the sizes of the hyperedges (and not simply the number of hyperedges). Our results show that for a feasible instance of DS-Hypergraph-CA-using-H, there exists a solution hypergraph in which the number of hyperedges is linear. Consequently, the size of such a solution hypergraph is quadratic in the number of vertices. However, the following structural question is open: for feasible instances of DS-Hypergraph-CA-using-H, does there exist a solution hypergraph whose *size* is linear in the number of vertices? We believe that answering this question would be a helpful stepping stone towards answering the previous algorithmic question.

3. The recent fast algorithms for DS-Graph-CA-using-E [24, 25, 26] rely on the fast construction of the family of *extreme sets* for the graph cut function. These fast construction algorithms heavily leverage the fact that the extreme set family is laminar. We note that the family of extreme sets is laminar for any *posimodular* function [90]. The hypergraph cut function is symmetric and submodular, and thus also posimodular. Is it possible to compute the extreme set family of hypergraphs in near-linear time?

4 Vertex Deletion to Reduce Density

In this section, we describe our results from ongoing work on GraphDD and SupmodDD, and also results for FVS and PFDS problems which are journal submission [30].

4.1 Density Deletion Set

We recall that the *density* of a graph $G = (V, E)$ is defined as $\lambda^*_G := \max_{S \subseteq V} \frac{|E[S]|}{|S|}$. We also recall the GraphDD problem which will be the subject of this section: the input is a graph $G = (V, E)$, vertex deletion costs $c : V \to \mathbb{R}_+$, and a density target ρ. The goal is to compute a set of vertices $F \subseteq V$ of minimum cost $c(F)$ such that $\lambda^*_{G-F} \leq \rho$. When the target density ρ is fixed, we refer to the problem as ρ-GraphDD

Approximation Hardness. Our first result shows that ρ-GraphDD is $o(\log n)$-inapproximable for all $\rho > 1$.

Theorem 4.1 (Ongoing Work)**.** *For every constant $\rho > 1$, there is no $o(\log n)$-approximation for ρ-GraphDD assuming $P \neq NP$, where n is the number of vertices in the input instance.*

We prove Theorem 4.1 via an approximation preserving reduction from the SetCover problem. In light of connections between GraphDD and various other problems, this approximation hardness is somewhat surprising. In fact, Theorem 4.1 basically resolves the approximability of ρ-GraphDD and shows that the problem exhibits a sharp *phase transition*: it admits a 2-approximation for $\rho \leq 1$ (via Fujito's results [52]) and becomes $o(\log n)$-inapproximable for $\rho > 1$.

Bicriteria Approximation. Our next result is a bicriteria approximation for ρ-GraphDD .

Theorem 4.2 (Ongoing Work)**.** *There exists a polynomial time algorithm which takes as input a graph $G = (V, E)$, vertex deletion costs $c : V \to \mathbb{R}_{\geq 0}$, a target density $\rho \in \mathbb{R}$, and an error parameter $\epsilon \in (0, 1/2)$, and returns a set $S \subseteq V$ such that:*

1. $\lambda^*_{G-S} \leq \left(\frac{1}{1-2\epsilon}\right) \cdot \rho$,
2. $\sum_{u \in S} c_u \leq \left(\frac{1}{\epsilon}\right) \cdot \texttt{OPT}$,

where $\texttt{OPT}$ denotes the cost of an optimum solution to ρ-GraphDD on the instance (G, c).

We prove Theorem 4.2 by rounding the LP relaxation of a new ILP formulation for GraphDD based on the connections between density and fractional orientations—we omit the details of this LP here for brevity (but we elaborate more on this connection in Section 4.3.1). A somewhat surprising phenomenon here is that while we were able to get a good bicriteria result from rounding this LP, we were able to show that the integrality gap of this LP is actually $\Omega(n)$ even for 2-GraphDD.

Remark 4.1. *Since we have a bicriteria approximation algorithm for GraphDD by Theorem 4.2, it is natural to wonder whether this would imply anything for SetCover via the reduction mentioned in the remark above. Unfortunately, our reduction is quite brittle from a bicriteria standpoint—the density of the GraphDD instance constructed via the reduction is already very close to what we set to be the target density. In particular, the empty set would be a valid bicriteria-approximate solution, thereby giving us no information about the set cover instance.*

4.2 Supermodular Density Deletion Set

The *density* of the supermodular function f is defined as $\lambda^*_f := \max_{S \subseteq V} \frac{f(S)}{|S|}$. We define the SupmodDD problem below which naturally generalizes the GraphDD problem.

Definition 4.1 (SupmodDD)**.** *The problem is defined as follows:*

Input:	*Supermodular function* $f : 2^V \to \mathbb{Z}$,
	Element deletion costs $c : V \to \mathbb{R}_+$, *and*
	Positive integer $\rho \in \mathbb{Z}_+$.
Goal:	*Compute* $\arg\min\{c(F) : F \subseteq V$ *and* $\lambda^*_{f\|_{V-F}} \leq \rho\}$.

Approximability. For a function $f : 2^V \to \mathbb{R}$, we define the marginal $f(v|S) := f(S + v) - f(S)$ for every $v \in V$ and $S \subseteq V$. Our first result shows that there is a polynomial time approximation algorithm for ρ-SupmodDD when ρ is integer-valued, where the approximation factor depends on the maximum marginal value of the input function.

Theorem 4.3 (Ongoing work)**.** *ρ-SupmodDD for input function $f : 2^V \to \mathbb{Z}$ and integer-valued ρ admits a polynomial time $O(\log(\max_{v\in V} f(v|V-v)))$-approximation algorithm.*

We show Theorem 4.3 via a reduction to SubmodCover. As part of this proposal, we also aim explore this connection between SubmodCover and ρ-SupmodDD further. In fact, in going work, we also prove that SubmodCover reduces to 1-SupmodDD, thus essentially showing the equivalence of SubmodCover and SupmodDD. We believe that it is useful to have this equivalence explicitly known given that vertex deletion problems arise naturally but seem different from covering problems on first glance.

Bicriteria Approximation. Our next result is a bicriteria algorithm for SupmodDD. Unlike the case of graphs, it is not clear how to write an integer programming formulation for SupmodDD whose LP-relaxation is polynomial-time solvable. Instead, we take inspiration from the very recent work of [113] on vertex deletion to reduce treewidth. We design a combinatorial randomized algorithm that yields a bi-criteria approximation for SupmodDD, where the bicriteria approximation bounds are based on a parameter c_f that depends on the input supermodular function f. The parameter c_f was defined in a recent work on DSS to unify the analysis of the greedy peeling algorithm [35]. We now define the parameter.

Definition 4.2 ([35])**.** *For a normalized supermodular function $f : 2^V \to \mathbb{R}_{\geq 0}$, we define*

$$c_f := \max\left\{\frac{\sum_{u\in S} f(u|S-u)}{f(S)} : S \subseteq V\right\}.$$

We note that $1 \leq c_f \leq |V|$ and moreover, $c_f = 1$ if and only if the function f is modular. If f is the induced edge function of a graph (i.e., $f(S) := |E(S)|$ for every $S \subseteq V$ where $G = (V, E)$ is a graph), then $c_f \leq 2$. This follows from the observation that the sum of degrees is at most twice the number of edges in a graph. Similarly, if f is the induced edge function of a hypergraph with rank r (i.e., all hyperedges have size at most r), then $c_f \leq r$. We refer the reader to [35, 110] for bounds on the parameter c_f for the function f arising in the non-trivial setting of p-mean densest subgraph when $p \geq 1$. We show the following bicriteria approximation for SupmodDD.

Theorem 4.4 (Ongoing work)**.** *There exists a randomized polynomial time algorithm which takes as input a normalized, monotone supermodular function $f : 2^V \to \mathbb{Z}$ (given by oracle access), element costs $c : V \to \mathbb{R}_+$, a target density $\rho \in \mathbb{R}$, and an error parameter $\epsilon \in (0, 1)$, and returns a set $S \subseteq V$ such that:*

1. $\lambda^*_{f|_{V-S}} \leq c_f(1+\epsilon)\cdot\rho$,
2. $\mathbb{E}[c(S)] \leq c_f(1 + 1/\epsilon)\cdot \texttt{OPT}$,

where `OPT` denotes the cost of an optimal ρ-SupmodDD.

Our algorithm to prove Theorem 4.4 is based on the following simple idea. Suppose that we had the following two capabilities: (1) a preprocessing step that could modify any input function so that it satisfies additional properties without changing the cost of the optimal solution, and (2) a way to assign non-negative *potentials* $\pi : V \to \mathbb{R}_{\geq 0}$ to the elements of the ground set of the modified function such that the potential value $\pi(X)$ of optimal solution X is large, say at least $\alpha \cdot \pi(V)$. Then, a natural algorithm—at least when the vertex deletion costs are uniform—would be to exhaustively perform the following two steps: do the preprocessing of step (1), compute the potentials of step (2), sample a vertex in proportion to the potentials, define a residual instance and repeat. This would ensure an α-approximation for the problem in expectation via a martingale argument. Unfortunately, the hardness result from Theorem 4.1 suggests that we are unlikely to obtain good vertex potentials for ρ-SupmodDD in general. However, we leverage supermodularity and show that if the density of the input function is at least β times the target density, then we can indeed find such good vertex potentials. This gives us an (α, β)-bicriteria guarantee, where the values α and β are as given in Theorem 4.4. The preprocessing step we use is a truncation of the *dense decomposition* [63] for the supermodular function, while the potential values we use are based on the *marginal gains* of the supermodular function [35]. Moreover, since the cost function to delete elements may be arbitrary we sample a vertex u in proportion its potential-to-cost ratio, i.e. $\pi(u)/c(u)$.

4.3 Pseudoforest Deletion Set

Preliminaries. A graph $G = (V, E)$ is a 2-*pseudotree* if it is connected and has $|E| \geq |V| + 1$ (i.e., the graph has at least 2 edges in addition to a spanning tree). We will denote the following polyhedra as weak density polyhedron and 2-pseudotree cover polyhedron respectively, and the constraints describing them as weak density constraints and 2-pseudotree cover constraints respectively:

$$P_{\text{WD}}(G) := \left\{x \in \mathbb{R}^V_{\geq 0} : \sum\nolimits_{u \in S} (d_S(u) - 1)x_u \geq |E[S]| - |S| \ \forall S \subseteq V\right\}, \text{ and} \tag{1}$$

$$P_{\text{2-PT-cover}}(G) := \left\{x \in \mathbb{R}^V_{\geq 0} : \sum\nolimits_{u \in U} x_u \geq 1 \ \forall U \subseteq V \text{ such that } G[U] \text{ contains a 2-pseudotree}\right\}. \tag{2}$$

Weak density constraints and 2-pseudotree covering constraints are valid for PFDS. In particular, the following are ILP formulations for PFDS:

$$\min\left\{\sum\nolimits_{u \in V} c_u x_u : \ x \in P_{\text{WD}}(G) \cap \mathbb{Z}^V\right\} \text{ and} \tag{PFDS-IP: WD}$$

$$\min\left\{\sum\nolimits_{u \in V} c_u x_u : \ x \in P_{\text{WD}}(G) \cap P_{\text{2-PT-cover}}(G) \cap \mathbb{Z}^V\right\}. \tag{PFDS-IP: WD-and-2PT-cover}$$

The local-ratio technique for PFDS due to Lin, Feng, Fu, and Wang [83] can be converted to a 2-approximation for PFDS with respect to the following LP-relaxation via the primal-dual technique:

$$\min\left\{\sum\nolimits_{u \in V} c_u x_u : \ x \in P_{\text{WD}}(G) \cap P_{\text{2-PT-cover}}(G)\right\}. \tag{PFDS-LP: WD-and-2PT-cover}$$

4.3.1 Poly-sized ILP Formulations and LP Integrality Gap

Our new ILP for PFDS is based on Charikar's LP for the densest subgraph problem (DSG) [32]. In DSG, the input is an undirected graph $G = (V, E)$ and the goal is to find an induced subgraph $G[S]$ of maximum density. Charikar formulated an LP to compute the density of a graph. The dual of Charikar's LP can be interpreted as a fractional orientation problem. Using that dual, we obtain an ILP for PFDS. We describe the details now. Via Charikar's LP and previous results, one can show that an unweighted graph G has density at most λ iff the edges of G can be fractionally oriented such that the total fractional in-degree at every vertex is at most λ. For an edge $e = uv$ we use variables $y_{e,u}$ and $y_{e,v}$ to denote the fractional amount of e that is oriented towards u and v respectively. We recall that in PFDS the goal is to remove vertices such that the residual graph has density at most 1. Thus, we also have variables x_u for each $u \in V$ to indicate whether u is deleted. An edge $e = uv$ is in the residual graph only if u and v are not deleted. These observations allow us to formulate the an ILP for PFDS based on the polyhedron below. We refer to this as the orientation polyhedron:

$$P_{\text{orient}}(G) := \left\{ (x, y) : \begin{array}{l} x_u + x_v + y_{e,u} + y_{e,v} \geq 1 \ \forall e \in E : e = uv \\ x_u + \sum_{e \in \delta(u)} y_{e,u} \leq 1 \ \forall u \in V \\ x_u \geq 0 \ \forall u \in V \\ y_{e,u} \geq 0 \ \forall e \in \delta(u), u \in V. \end{array} \right\}.$$

We will denote the projection of $P_{\text{orient}}(G)$ to the x variables by $Q_{\text{orient}}(G)$. For non-negative costs $c : V \to \mathbb{R}_{\geq 0}$, we consider the following formulations:

$$\min\left\{\sum\nolimits_{u \in V} c_u x_u : x \in Q_{\text{orient}}(G) \cap \mathbb{Z}^V\right\} \text{ and} \tag{PFDS-IP: orient}$$

$$\min\left\{\sum\nolimits_{u \in V} c_u x_u : x \in Q_{\text{orient}}(G) \cap P_{\text{2-PT-cover}}(G) \cap \mathbb{Z}^V\right\}. \tag{PFDS-IP: orient-and-2PT-cover}$$

We will be interested in the integrality gap of the following LP-relaxation of (PFDS-IP: orient-and-2PT-cover):

$$\min\left\{\sum_{u\in V} c_u x_u : x \in Q_{\text{orient}}(G) \cap P_{\text{2-PT-cover}}(G)\right\}. \qquad \text{(PFDS-LP: orient-and-2PT-cover)}$$

Theorem 4.5 ([30]). *For an input graph $G = (V, E)$ with non-negative costs $c : V \to \mathbb{R}_{\geq 0}$, (PFDS-IP: orient) and (PFDS-IP: orient-and-2PT-cover) are integer linear programming formulations for PFDS. Moreover, we have the following properties:*

(1) $Q_{orient}(G) \subseteq P_{WD}(G)$ for every graph G and there exist graphs G for which $Q_{orient}(G) \subsetneq P_{WD}(G)$.

(2) The LP (PFDS-LP: orient-and-2PT-cover) is solvable in polynomial time and its integrality gap is at most 2.

In the full paper [30], we show a polynomial-time separation oracle for the family of 2-pseudotree cover constraints via a reduction to steiner tree with constant number of terminals—this is a necessary step for showing Theorem 4.5(2). We also construct an instance which exhbits that the integrality gap upper bound mentioned in Theorem 4.5 is tight. As a consequence of Theorem 4.5, we immediately obtain an integer linear program for PFDS whose LP-relaxation is solvable in polynomial time and has integrality gap at most 2 — namely (PFDS-IP: orient-and-2PT-cover). We also bound the integrality gap of the LP-relaxation of (PFDS-IP: orient), but this will be based on extreme point properties and will be discussed next.

4.3.2 Extreme Point Properties

In this section, we present our extreme point results for polyhedra associated with PFDS. Our first result below shows an extreme point result for the weak density polyhedron.

Theorem 4.6 ([30]). *Let $G = (V, E)$ be a graph that is not a pseudoforest. For every extreme point x of the polyhedron $P_{WD}(G)$, there exists a vertex $u \in V$ such that $x_u \geq 1/3$.*

Our proof of Theorem 4.6 is based on a conditional supermodularity property—if all coordinates are small, then the weak density constraints have a supermodular property; we use this supermodular property to show the existence of a structured basis for the extreme point which is subsequently used to arrive at a contradiction. The conditional supermodularity property based proof has not previously appeared in the literature on iterated rounding, and may be of independent interest. We note that Theorem 4.6 immediately implies that the integrality gap of the natural LP relaxation of (PFDS-IP: WD) is at most 3 by the iterated rounding framework [103]. However, we note this is interesting only from a polyhedral viewpoint currently and is not of help from the perspective of algorithm design since implementing the iterative rounding procedure requires solving the LP-relaxation in polynomial time, but we do not know how to do this yet.

We next show that although $Q_{\text{orient}}(G)$ is a subset of $P_{\text{WD}}(G)$ (as shown in Theorem 4.5), the extreme point result for $P_{\text{WD}}(G)$ given in Theorem 4.6 still holds for $Q_{\text{orient}}(G)$. We will say that an extreme point of a polyhedron is *minimal* if for each variable, reducing the value of that variable by any $\epsilon > 0$ while keeping the rest of the variables unchanged results in a point that is outside the polyhedron. We note that extreme points of a polyhedron along non-negative objective directions will be minimal.

Theorem 4.7 ([30]). *Let $G = (V, E)$ be a graph that is not a pseudoforest. For every minimal extreme point (x, y) of the polyhedron $P_{orient}(G)$, there exists a vertex $u \in V$ such that $x_u \geq 1/3$.*

Our proof of Theorem 4.7 is purely based on combinatorial arguments that rely on careful edge counting. Similar to the previous Theorem 4.6, Theorem 4.7 immediately implies that the integrality gap of the natural LP relaxation of (PFDS-IP: orient) is at most 3. We conclude the section with the remark that in the full paper [30] we construct instances which exhibit that Theorem 4.6 and Theorem 4.7 are tight.

4.4 Feedback Vertex Set

Preliminaries. The following polyhedron will be referred to as the cycle cover polyhedron:

$$P_{\text{cycle-cover}}(G) := \left\{ x \in \mathbb{R}^V_{\geq 0} : \sum\nolimits_{u \in U} x_u \geq 1 \ \forall U \subseteq V \text{ such that } G[U] \text{ contains a cycle} \right\}. \tag{3}$$

We will denote the constraints describing the cycle cover polyhedron as cycle cover constraints. It is known that the integrality gap of the natural LP over this polyhedron for FVS is $\Theta(\log n)$ [8].

4.4.1 An intermediate LP

In this section, we formulate an intermediate ILP for FVS whose LP-relaxation has integrality gap at most 2, but it is unclear if this LP-relaxation is polynomial-time solvable. In the next section, we will formulate a *polynomial-sized* ILP for FVS whose LP-relaxation is at least as strong as that of the intermediate ILP, thereby achieving our goal of answering Question 10. The intermediate ILP will require the weak density polyhedron (1) from Section 4.3. In particular, for non-negative costs $c : V \to \mathbb{R}_{\geq 0}$, we consider the following formulation and its LP-relaxation:

$$\min \left\{ \sum\nolimits_{u \in V} c_u x_u : x \in P_{\text{WD}}(G) \cap P_{\text{cycle-cover}}(G) \cap \mathbb{Z}^V \right\} \quad \text{and} \qquad \text{(FVS-IP: WD-and-cycle-cover)}$$

$$\min \left\{ \sum\nolimits_{u \in V} c_u x_u : x \in P_{\text{WD}}(G) \cap P_{\text{cycle-cover}}(G) \right\}. \qquad \text{(FVS-LP: WD-and-cycle-cover)}$$

Our result below shows that (FVS-IP: WD-and-cycle-cover) is an ILP for FVS whose LP-relaxation (FVS-LP: WD-and-cycle-cover) has integrality gap at most 2.

Theorem 4.8 ([30]). *For an input graph $G = (V, E)$ with non-negative costs $c : V \to \mathbb{R}_{\geq 0}$,* (FVS-IP: WD-and-cycle-cover) *is an integer linear programming formulation for FVS. Moreover, the integrality gap of* (FVS-LP: WD-and-cycle-cover) *is at most* 2.

Our proof of the second part of Theorem 4.8 is by exhibiting a primal-dual algorithm that closely follows the algorithm of Chudak et. al. [39]. We note that we cannot solve (FVS-LP: WD-and-cycle-cover) since we do not have a polynomial-time separation oracle for the family of weak density constraints (although we do have a polynomial-time separation oracle for the family of cycle cover constraints). However, we use this integrality gap result in the next section to obtain a polynomial-sized LP with the same integrality gap.

Remark 4.2. *It is tempting to bound the integrality gap of (FVS-LP: WD-and-cycle-cover) by proving an extreme point result similar to Theorem 4.6 and Theorem 4.7. However, in the full paper [30], we construct an instance where there exists an extreme point optimum of (FVS-LP: WD-and-cycle-cover) all of whose coordinates have value at most* 1/3.

4.4.2 Poly-sized ILP Formulations and LP Integrality Gaps

In this section, we present the following main result that affirmatively answers Question 10.

Theorem 4.9 ([30]). *There exists a polynomial-sized integer linear programming formulation for FVS whose LP-relaxation has integrality gap at most* 2.

We prove Theorem 4.9 by showing three different polynomial-sized integer linear programs for FVS all of whose LP-relaxations have integrality gap at most 2:

1. The first formulation is

$$\min \left\{ \sum\nolimits_{u \in V} c_u x_u : x \in Q_{\text{orient}}(G) \cap P_{\text{cycle-cover}}(G) \cap \mathbb{Z}^V \right\}. \qquad \text{(FVS-IP: orient-and-cycle-cover)}$$

By Theorem 4.5(1), we have that $Q_{\text{orient}}(G) \subseteq P_{\text{WD}}(G)$. As a consequence of Theorem 4.8, the integrality gap of the following LP-relaxation is also at most 2:

$$\min \left\{ \sum_{u \in V} c_u x_u : x \in Q_{\text{orient}}(G) \cap P_{\text{cycle-cover}}(G) \right\}. \qquad \text{(FVS-LP: orient-and-cycle-cover)}$$

Both $Q_{\text{orient}}(G)$ and $P_{\text{cycle-cover}}(G)$ admit a polynomial-sized description—see the full paper [30] for a polynomial-sized description of $P_{\text{cycle-cover}}(G)$). Consequently, we have a polynomial-sized ILP for FVS whose LP-relaxation has integrality gap at most 2.

2. The second formulation is the Chekuri-Madan formulation who, as we remarked earlier in Section 1.3.1, formulated an ILP for Subset-FVS and showed that the integrality gap of its LP-relaxation is at most 13 [34]. We show that their LP-relaxation specialized for FVS has integrality gap at most 2 by proving that it is at least as strong as (FVS-LP: orient-and-cycle-cover). Our result gives additional impetus to improving the integrality gap of their LP-relaxation for Subset-FVS.

3. Our third formulation to prove Theorem 4.9 is based on the orientation perspective, but without cycle cover constraints (as opposed to our first formulation). Here, we give an orientation based ILP formulation whose associated polyhedron is contained in the *strong density polyhedron* considered by Chudak et. al. [39]. Since the integrality gap of the LP-relaxation over the strong desnity polyhedron is at most 2 (as shown by Chudak et. al. [39]), the integrality gap of our third formulation is also at most 2.

4.5 Open Problems

In this section, we list some open problems that arose as a result of our work.

1. Does there exist a parallel $O(1)$-approximation algorithm for FVS?

2. Does there exist a $(O(1), O(1))$-bicriteria algorithm for SUPMODDD? In particular, does there exist a polynomial time algorithm that returns a set that has at most $O(1)$ times the cost of the optimal ρ-SUPMODDD, and the density of the function obtained after deleting the returned set is $O(\rho)$?

3. Does the *strong density polyhedron* considered by Chudak et. al. for FVS have good extreme point structure? In our full paper [30], we formulate a concrete conjecture on the extreme point structure of this polyhedron. Resolving this conjecture would lead to alternative an LP-rounding algorithm for approximating FVS and potentially provide insights into improving the approximation ratio for the more general SUBSETFVS problem [34, 44, 45]. Improving the approximation ratio of SUBSETFVS (either lower or upper bound) is also an open problem.

4. One of our main result shows that the integrality gap of the Chekuri-Madan LP [34], when specialized for FVS, is at most 2. What is the integrality gap of the LP for the general SUBSETFVS problem? In the same work, Chekuri and Madan also formulate an LP for the SUBSETFEEDBACKEDGESET problem. Resolving the integrality gap of this LP is also an open problem.

References

[1] Head-disjoint strongly connected orientations, http://lemon.cs.elte.hu/egres/open/Head-disjoint_strongly_connected_orientations.

[2] O. Alrabiah, V. Guruswami, and R. Li. Randomly punctured Reed–Solomon codes achieve list-decoding capacity over linear-sized fields. Preprint on arXiv: 2304.09445, 2023.

[3] V. Bafna, P. Berman, and T. Fujito. Constant ratio approximations of the weighted feedback vertex set problem for undirected graphs. In *Algorithms and Computations*, pages 142–151, 1995.

[4] J. Bang-Jensen, A. Frank, and B. Jackson. Preserving and increasing local edge-connectivity in mixed graphs. *SIAM Journal on Discrete Mathematics*, 8(2):155–178, 1995.

[5] J. Bang-Jensen, H. N. Gabow, T. Jordan, and Z. Szigeti. Edge-connectivity augmentation with partition constraints. Technical report, 1997.

[6] J. Bang-Jenson and B. Jackson. Augmenting hypergraphs by edges of size two. *Mathematical Programming*, 84:467–481, 1999.

[7] N. Bansal, O. Svensson, and L. Trevisan. New notions and constructions of sparsification for graphs and hypergraphs. In *IEEE 60th Annual Symposium on Foundations of Computer Science*, FOCS, pages 910–928, 2019.

[8] R. Bar-Yehuda, D. Geiger, J. Naor, and R. M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and bayesian inference. *SIAM Journal on Computing*, 27(4):942–959, 1998.

[9] C. Bazgan, A. Nichterlein, and S. Vazquez Alferez. Destroying Densest Subgraphs Is Hard. In *19th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2024)*, pages 6:1–6:17, 2024.

[10] A. Becker and D. Geiger. Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artificial Intelligence*, 83:167–188, 1996.

[11] A. Benczúr. Parallel and fast sequential algorithms for undirected edge connectivity augmentation. *Math. Program.*, 84:595–640, 1999.

[12] A. Benczúr and D. Karger. Augmenting Undirected Edge Connectivity in $\tilde{O}(n^2)$ Time. *Journal of Algorithms*, 37(1):2–36, 2000.

[13] A. Benczúr and M. Goemans. *Deformable Polygon Representation and Near-Mincuts*, page 103–135. Springer, Berlin, Heidelberg, 2008.

[14] K. Bérczi, K. Chandrasekaran, C. Chekuri, T. Király, and S. Kulkarni. Private communication.

[15] K. Bérczi, K. Chandrasekaran, T. Király, and S. Kulkarni. Splitting-Off in Hypergraphs. In *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 23:1–23:20, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[16] A. Bernáth, R. Grappe, and Z. Szigeti. Partition constrained covering of a symmetric crossing supermodular function by a graph. *SIAM Journal on Discrete Mathematics*, 31(1):335–382, 2017.

[17] A. Bernáth and T. Király. Covering skew-supermodular functions by hypergraphs of minimum total size. *Operations Research Letters*, 37(5):345–350, 2009.

[18] A. Bernáth and T. Király. A unifying approach to splitting-off. *Combinatorica*, 32:373—401, 2012.

[19] A. Bhalgat, R. Hariharan, T. Kavitha, and D. Panigrahi. Fast Edge Splitting and Edmonds' Arborescence Construction for Unweighted Graphs. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 455–464, 2008.

[20] J. Blauth and M. Nägele. An improved approximation guarantee for Prize-Collecting TSP. Preprint on arXiv: 2212.03776, 2023.

[21] J. Byrka, F. Grandoni, and A. J. Ameli. Breaching the 2-approximation barrier for connectivity augmentation: a reduction to steiner tree. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 815–825, 2020.

[22] K. Bérczi, K. Chandrasekaran, T. Király, and S. Kulkarni. Hypergraph connectivity augmentation in strongly-polynomial time. In T. Chan, F. Johannes, J. Iacono, and G. Herman, editors, *32nd Annual European Symposium on Algorithms (ESA 2024)*, volume 308 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Royal Holloway, London, United Kingdom, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[23] G.-R. Cai and Y.-G. Sun. The minimum augmentation ofany graph to k-edge-connected graph. *Networks*, 19:151–172, 1989.

[24] R. Cen, W. He, J. Li, and D. Panigrahi. Steiner connectivity augmentation and splitting-off in poly-logarithmic maximum flows. In *Proceedings of the 34th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 2449–2488, 2023.

[25] R. Cen, J. Li, and D. Panigrahi. Augmenting edge connectivity via isolating cuts. In *Proceedings of the 33rd Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 3237–3252, 2022.

[26] R. Cen, J. Li, and D. Panigrahi. Edge connectivity augmentation in near-linear time. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC, pages 137–150, 2022.

[27] Y. H. Chan, W. S. Fung, L. C. Lau, and C. K. Yung. Degree Bounded Network Design with Metric Costs. *SIAM Journal on Computing*, 40(4):953–980, 2011.

[28] K. Chandrasekaran and C. Chekuri. Hypergraph k-cut for fixed k in deterministic polynomial time. *Mathematics of Operations Research*, 47(4), 2022. Prelim. version in FOCS 2020.

[29] K. Chandrasekaran and C. Chekuri. Min-max partitioning of hypergraphs and symmetric submodular functions. *Combinatorica*, 43:455–477, 2023. Prelim. version in SODA 2021.

[30] K. Chandrasekaran, C. Chekuri, S. Fiorini, S. Kulkarni, and S. Weltge. Polyhedral aspects of feedback vertex set and pseudoforest deletion set, 2024.

[31] K. Chandrasekaran, C. Xu, and X. Yu. Hypergraph k-Cut in randomized polynomial time. *Mathematical Programming*, 186:85–113, 2021. Prelim. version in SODA 2018.

[32] M. Charikar. Greedy Approximation Algorithms for Finding Dense Components in a Graph. In *Approximation Algorithms for Combinatorial Optimization*, pages 84–95, 2000.

[33] C. Chekuri and N. Korula. A Graph Reduction Step Preserving Element-Connectivity and Packing Steiner Trees and Forests. *SIAM Journal on Discrete Mathematics*, 28(2):577–597, 2014.

[34] C. Chekuri and V. Madan. Constant factor approximation for subset feedback set problems via a new LP relaxation. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 808–820, 2016.

[35] C. Chekuri, K. Quanrud, and M. R. Torres. Densest subgraph: Supermodularity, iterative peeling, and flow. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1531–1555. SIAM, 2022.

[36] C. Chekuri and B. Shepherd. Approximate Integer Decompositions for Undirected Network Design Problems. *SIAM J. Discrete Math.*, 23:163–177, 2008.

[37] C. Chekuri and C. Xu. Minimum cuts and sparsification in hypergraphs. *SIAM Journal on Computing*, 47(6):2118–2156, 2018.

[38] Y. Chen, S. Khanna, and A. Nagda. Near-linear size hypergraph cut sparsifiers. In *IEEE 61st Annual Symposium on Foundations of Computer Science*, FOCS, pages 61–72, 2020.

[39] F. A. Chudak, M. X. Goemans, D. S. Hochbaum, and D. P. Williamson. A primal–dual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs. *Operations Research Letters*, 22(4):111–118, 1998.

[40] B. Cosh. *Vertex splitting and connectivity augmentation in hypergraphs.* PhD thesis, University of London, 2000.

[41] B. Cosh, B. Jackson, and Z. Király. Local edge-connectivity augmentation in hypergraphs is NP-complete. *Discrete Applied Mathematics*, 158(6):723–727, 2010.

[42] W. H. Cunningham. Optimal attack and reinforcement of a network. *Journal of the ACM (JACM)*, 32(3):549–561, 1985.

[43] P. Erdös and L. Pósa. On the maximal number of disjoint circuits of a graph. *Publ. Math. Debrecen*, 9:3–12, 1962.

[44] G. Even, J. Naor, B. Schieber, and L. Zosin. Approximating minimum subset feedback sets in undirected graphs with applications. *SIAM Journal on Discrete Mathematics*, 13(2):255–267, 2000.

[45] G. Even, J. Naor, and L. Zosin. An 8-approximation algorithm for the subset feedback vertex set problem. *SIAM Journal on Computing*, 30(4):1231–1252, 2000.

[46] K. Fox, D. Panigrahi, and F. Zhang. Minimum cut and minimum k-cut in hypergraphs via branching contractions. *ACM Trans. Algorithms*, 19(2), 2023. Prelim. version in SODA 2019.

[47] A. Frank. Augmenting graphs to meet edge-connectivity requirements. *SIAM Journal on Discrete Mathematics*, 5(1):25–53, 1992.

[48] A. Frank. Applications of submodular functions. *Surveys in Combinatorics, 1993 (Keele)*, pages 85–136, 1993.

[49] A. Frank. Connectivity augmentation problems in network design. *Mathematical Programming: State of the Art 1994*, pages 34–63, 1994.

[50] A. Frank. *Connections in combinatorial optimization.* Oxford University Press, Oxford, 2011.

[51] A. Frank and Z. Király. Graph orientations with edge-connection and parity constraints. *Combinatorica*, 22:47–70, 2002.

[52] T. Fujito. Approximating node-deletion problems for matroidal properties. *Journal of Algorithms*, 31(1):211–227, 1999.

[53] H. N. Gabow. Efficient splitting off algorithms for graphs. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, STOC, pages 696—705, 1994.

[54] M. Ghaffari, D. Karger, and D. Panigrahi. Random Contractions and Sampling for Hypergraph and Hedge Connectivity. In *Proceedings of the 28th annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 1101–1114, 2017.

[55] A. Gionis and C. E. Tsourakakis. Dense subgraph discovery: Kdd 2015 tutorial. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 2313–2314, New York, NY, USA, 2015. Association for Computing Machinery.

[56] M. Goemans. Approximate Edge Splitting. *SIAM Journal on Discrete Mathematics*, 14(1):138–141, 2001.

[57] M. Goemans and D. Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Mathematical Programming*, 60:145–166, 1993.

[58] A. V. Goldberg. *Finding a maximum density subgraph.* University of California Berkeley, 1984.

[59] F. Grandoni, A. J. Ameli, and V. Traub. Breaching the 2-approximation barrier for the forest augmentation problem. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 1598–1611, 2022.

[60] Z. Guo, R. Li, C. Shangguan, I. Tamo, and M. Wootters. Improved List-Decodability and List-Recoverability of Reed-Solomon Codes via Tree Packings. In *IEEE 62nd Annual Symposium on Foundations of Computer Science*, FOCS, pages 708–719, 2022.

[61] M. M. Halldórsson, T. Ishii, K. Makino, and K. Takazawa. Posimodular function optimization. *Algorithmica*, 84(4):1107–1131, 2022.

[62] E. Harb, K. Quanrud, and C. Chekuri. Faster and scalable algorithms for densest subgraph and decomposition. *Advances in Neural Information Processing Systems*, 35:26966–26979, Dec. 2022.

[63] E. Harb, K. Quanrud, and C. Chekuri. Faster and scalable algorithms for densest subgraph and decomposition. In *Advances in Neural Information Processing Systems*, 2022.

[64] R. Hariharan, T. Kavitha, D. Panigrahi, and A. Bhalgat. An $\tilde{\mathrm{O}}(\mathrm{mn})$ gomory-hu tree construction algorithm for unweighted graphs. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, page 605–614, New York, NY, USA, 2007. Association for Computing Machinery.

[65] M. Henzinger and D. Williamson. On the number of small cuts in a graph. *Information Processing Letters*, 59:41–44, 1996.

[66] H. R. Hind and O. Oellermann. Menger-type results for three or more vertices. *Congressus Numerantium*, pages 179–204, 1996.

[67] T. Ishii and M. Hagiwara. Minimum augmentation of local edge-connectivity between vertices and vertex subsets in undirected graphs. *Discrete Applied Mathematics*, 154(16):2307–2329, 2006.

[68] K. Jain, M. Mahdian, and M. R. Salavatipour. Packing Steiner Trees. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 266–274, 2003.

[69] A. Jambulapati, Y. P. Liu, and A. Sidford. Chaining, group leverage score overestimates, and fast spectral hypergraph sparsification. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC, pages 196–206, 2023.

[70] T. Jordán. On minimally k-edge-connected graphs and shortest k-edge-connected Steiner networks. *Discrete Applied Mathematics*, 131(2):421–432, 2003.

[71] M. Kapralov, R. Krauthgamer, J. Tardos, and Y. Yoshida. Towards tight bounds for spectral sparsification of hypergraphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC, pages 598–611, 2021.

[72] M. Kapralov, R. Krauthgamer, J. Tardos, and Y. Yoshida. Spectral Hypergraph Sparsifiers of Nearly Linear Size. In *IEEE 62nd Annual Symposium on Foundations of Computer Science*, FOCS, pages 1159–1170, 2022.

[73] S. Khuller and B. Saha. On finding dense subgraphs. In S. Albers, A. Marchetti-Spaccamela, Y. Matias, S. Nikoletseas, and W. Thomas, editors, *Automata, Languages and Programming*, pages 597–608, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[74] T. Király and L. C. Lau. Approximate min–max theorems for Steiner rooted-orientations of graphs and hypergraphs. *Journal of Combinatorial Theory, Series B*, 98:1233–1252, 11 2008. Preliminary version in FOCS 2006.

[75] D. Kogan and R. Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS, pages 367–376, 2015.

[76] M. Kriesell. Edge-disjoint trees containing some given vertices in a graph. *J. Comb. Theory Ser. B*, 88:53–63, 2003.

[77] T. Lanciano, A. Miyauchi, A. Fazzone, and F. Bonchi. A survey on the densest subgraph problem and its variants. *ACM Comput. Surv.*, 56(8), Apr. 2024.

[78] L. C. Lau. An Approximate Max-Steiner-Tree-Packing Min-Steiner-Cut Theorem. *Combinatorica*, 27:71–90, 2007.

[79] L. C. Lau and C. K. Yung. Efficient Edge Splitting-Off Algorithms Maintaining All-Pairs Edge-Connectivities. *SIAM Journal on Computing*, 42(3):1185–1200, 2013.

[80] J. R. Lee. Spectral hypergraph sparsification via chaining. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC, pages 207–218, 2023.

[81] J. Li and D. Panigrahi. Approximate gomory–hu tree is faster than $\boldsymbol{n-1}$ maximum flows. *SIAM Journal on Computing*, 53(4):1162–1180, 2024.

[82] P. Li and O. Milenkovic. Inhomogeneous hypergraph clustering with applications. *Advances in neural information processing systems*, 30, 2017.

[83] M. Lin, Q. Feng, B. Fu, and J. Wang. An approximation algorithm for the ℓ-pseudoforest deletion problem. *Theoretical Computer Science*, 806, 08 2019.

[84] L. Lovász. Lecture. *Presented in a Conference on Graph Theory, Prague*, 1974.

[85] L. Lovász. On some connectivity properties of Eulerian graphs. *Acta Math. Hungarica*, 28:129–138, 1976.

[86] L. Lovász. *Combinatorial Problems and Exercises, Second Edition.* American Mathematical Soc., 1993. First Edition: North-Holland, Amsterdam, 1979.

[87] W. Mader. A reduction method for edge-connectivity in graphs. In *Annals of Discrete Mathematics*, volume 3, pages 145–164. Elsevier, 1978.

[88] H. Miwa and H. Ito. NA-edge-connectivity augmentation problems by adding edges. *J. Oper. Res. Soc Japan*, 47:224–243, 2004.

[89] B. Monien and R. Schulz. Four approximation algorithms for the feedback vertex set problem. In *WG*, pages 315–326, 1981.

[90] H. Nagamochi and T. Ibaraki. *Algorithmic Aspects of Graph Connectivity.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2008.

[91] H. Nagamochi, K. Nishimura, and T. Ibaraki. Computing all small cuts in an undirected network. *SIAM Journal on Discrete Mathematics*, 10(3):469–481, 1997.

[92] M. Nägele and R. Zenklusen. A new contraction technique with applications to congruency-constrained cuts. *Mathematical Programming*, 183(2):455–481, 2020. Prelim. version in IPCO 2019.

[93] D. Naor, D. Gusfield, and C. Martel. A fast algorithm for optimally increasing the edge connectivity. *SIAM Journal on Computing*, 26(4):1139–1165, 1997.

[94] C. S. J. A. Nash-Williams. On orientations, connectivity and odd vertex pairings in finite graphs. *Canad. J. Math*, 12:555–567, 1960.

[95] M. Nägele and R. Zenklusen. A new contraction technique with applications to congruency-constrained cuts. *Mathematical Programming*, 183(1-2):455 – 481, 2020. Cited by: 8.

[96] S. Ornes. How big data carried graph theory into new dimensions. *Quanta Magazine*, 2021. https://www.quantamagazine.org/how-big-data-carried-graph-theory-into-new-dimensions-20210819/.

[97] G. Philip, A. Rai, and S. Saurabh. Generalized pseudoforest deletion: Algorithms and uniform kernel. *SIAM Journal on Discrete Mathematics*, 32(2):882–901, 2018.

[98] J.-C. Picard and M. Queyranne. A network flow solution to some nonlinear 0-1 programming problems, with applications to graph theory. *Networks*, 12(2):141–159, 1982.

[99] K. Quanrud. Quotient sparsification for submodular functions. Manuscript available at kentquanrud.com, November 2022.

[100] H. Reinstädtler, C. Schulz, and B. Uçar. Engineering edge orientation algorithms. In *DROPS-IDN/v2/document/10.4230/LIPIcs.ESA.2024.97*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024.

[101] H. E. Robbins. A Theorem on Graphs with an Application to a Problem of Traffic Control. *Amer. Math. Monthly*, 46:281–283, 1939.

[102] S. Schlag, T. Heuer, L. Gottesbüren, Y. Akhremtsev, C. Schulz, and P. Sanders. High-quality hypergraph partitioning. *ACM Journal of Experimental Algorithmics*, 27:1–39, 2023.

[103] M. Singh, R. Ravi, and L. C. Lau. *Iterative Methods in Combinatorial Optimization*. Cambridge University Press, 2011.

[104] T. Soma and Y. Yoshida. Spectral sparsification of hypergraphs. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 2570–2581, 2019.

[105] P. Sukprasert, Q. C. Liu, L. Dhulipala, and J. Shun. *Practical Parallel Algorithms for Near-Optimal Densest Subgraphs on Massive Graphs*, page 59–73. Proceedings. Society for Industrial and Applied Mathematics, Jan. 2024.

[106] Z. Szigeti. Hypergraph connectivity augmentation. *Math. Program.*, 84(3):519–527, 1999.

[107] V. Traub and R. Zenklusen. A better-than-2 approximation for weighted tree augmentation. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–12, 2022.

[108] V. Traub and R. Zenklusen. Local search for weighted tree augmentation and steiner tree. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3253–3272, 2022.

[109] V. Traub and R. Zenklusen. A $(1.5 + \epsilon)$-Approximation Algorithm for Weighted Connectivity Augmentation. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, page 1820–1833, 2023.

[110] N. Veldt, A. R. Benson, and J. Kleinberg. The generalized mean densest subgraph problem. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1604–1614, 2021.

[111] N. Veldt, A. R. Benson, and J. Kleinberg. Hypergraph cuts with general splitting functions. *SIAM Review*, 64(3):650–685, 2022.

[112] T. Watanabe and A. Nakamura. Edge-connectivity augmentation problems. *Journal of Computer and System Sciences*, 35(1):96–144, 1987.

[113] M. Włodarczyk. Losing treewidth in the presence of weights. In *(To appear) Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2025.