Densest Subgraph: Supermodularity, Iterative Peeling, and Flow

Chandra Chekuri* Kent Quanrud† Manuel R. Torres‡

Abstract

The densest subgraph problem in a graph (DSG), in the simplest form, is the following. Given an undirected graph $G = (V, E)$ find a subset $S \subseteq V$ of vertices that maximizes the ratio $|E(S)|/|S|$ where $E(S)$ is the set of edges with both endpoints in S. DSG and several of its variants are well-studied in theory and practice and have many applications in data mining and network analysis. In this paper we study fast algorithms and structural aspects of DSG via the lens of *supermodularity*. For this we consider the densest supermodular subset problem (DSS): given a non-negative supermodular function $f : 2^V \to \mathbb{R}_+$, maximize $f(S)/|S|$.

For DSG we describe a simple flow-based algorithm that outputs a $(1-\epsilon)$-approximation in deterministic $\tilde{O}(m/\epsilon)$ time where m is the number of edges. Our algorithm is the first to have a near-linear dependence on m and $1/\epsilon$ and improves previous methods based on an LP relaxation. It generalizes to hypergraphs, and also yields a faster algorithm for directed DSG.

Greedy peeling algorithms have been very popular for DSG and several variants due to their efficiency, empirical performance, and worst-case approximation guarantees. We describe a simple peeling algorithm for DSS and analyze its approximation guarantee in a fashion that unifies several existing results. Boob et al. [12] developed an *iterative* peeling algorithm for DSG which appears to work very well in practice, and made a conjecture about its convergence to optimality. We affirmatively answer their conjecture, and in fact prove that a natural generalization of their algorithm converges to a $(1-\epsilon)$-approximation for *any* supermodular function f; the key to our proof is to consider an LP formulation that is derived via the Lovász extension of a supermodular function. For DSG the bound on the number of iterations we prove is $O(\frac{\Delta \ln |V|}{\lambda^*} \cdot \frac{1}{\epsilon^2})$ where Δ is the maximum degree and λ^* is the optimum value. Our work suggests that iterative peeling can be an effective heuristic for several objectives considered in the literature.

Finally, we show that the 2-approximation for densest-at-least-k subgraph [37] extends to the supermodular setting. We also give a unified analysis of the peeling algorithm for this problem, and via this analysis derive an approximation guarantee for a generalization of DSS to maximize $f(S)/g(|S|)$ for a concave function g.

1 Introduction

The densest subgraph problem (DSG) is a canonical problem in network analysis and graph mining with several real-world applications. In its simplest form, the input to DSG is an undirected graph $G = (V, E)$ and the goal is to find a subset of vertices $S \subseteq V$ to maximize the average degree in the induced graph $G[S]$; equivalently, find S to maximize $\frac{|E(S)|}{|S|}$ where $E(S)$ is the set of edges with both endpoints in S. Dense subgraphs are of interest in a variety of application areas where such subgraphs reveal useful information. Although DSG is a popular and well-studied problem, the notion of "density" can vary by application, and researchers study several related problems in the broad line of work called dense subgraph discovery. We refer the reader to some surveys and tutorials [23, 28, 41] including a very recent one [63], and some recent papers [12, 13, 27, 43, 54, 65] and pointers therein. Constrained versions of DSG such as the densest k-subgraph problem (DkSG) (see [9, 25, 36, 39, 45] and pointers therein), densest *at most* k-subgraph problem (DamkSG) [3, 37], and the densest *at least* k-subgraph problem (DalkSG) [3, 37, 46] are also extensively studied. Our main focus is on DSG, and we discuss the constrained versions later.

*Dept. of Computer Science, Univ. of Illinois, Urbana-Champaign, Urbana, IL 61801. chekuri@illinois.edu. Supported in part by NSF grant CCF-1910149.

†Dept. of Computer Science, Purdue University, West Lafayette, IN 47907. krq@purdue.edu. Supported in part by NSF grant CCF-2129816.

‡Dept. of Computer Science, Univ. of Illinois, Urbana-Champaign, Urbana, IL 61801. manuelt2@illinois.edu. Supported in part by fellowships from NSF and the Sloan Foundation, and NSF grant CCF-1910149.

An attractive aspect of DSG is that it is a polynomial-time solvable problem with interesting connections to fundamental problems in combinatorial optimization. This can be exploited in algorithms and applications, and some of these insights help develop heuristics and approximation algorithms. It is helpful to consider a generalization of DSG that is tractable. A real-valued set function $f : 2^V \to \mathbb{R}$ is *supermodular* iff $f(A)+f(B) \leq f(A \cup B)+f(A \cap B)$ for all $A, B \subseteq V$. Equivalently, f is supermodular iff $f(v \mid A) \geq f(v \mid B)$ for all $B \subseteq A$ and $v \in V \setminus A$ where we let $f(v \mid S) \coloneqq f(S \cup \{v\}) - f(S)$ denote the marginal value of v with respect to S. A set function g is *submodular* iff the function $-g$ is supermodular (one can also define submodularity directly). Consider the following problem that we refer to as the generalized densest supermodular subset problem (GDSS): given a ground set V, a non-negative supermodular function $f : 2^V \to \mathbb{R}_+$, and a non-negative submodular function $g : 2^V \to \mathbb{R}_+$, find $S \subseteq V$ to maximize $\frac{f(S)}{g(S)}$. We use DSS (densest supermodular subset problem) to refer to the special case when $g(S) = |S|$ which will be the main focus of this paper[1]. It is well-known and easy to see that DSG is a special case of DSS; the function $f : 2^V \to \mathbb{R}_+$ where $f(S) = |E(S)|$ is supermodular. To see that GDSS is tractable, consider the decision problem: given f, g and $\lambda > 0$, is there an $S \subseteq V$ such that $f(S)/g(S) \geq \lambda$; equivalently, is there an S such that $\lambda g(S) - f(S) \leq 0$. This reduces to submodular set function minimization which admits a (strongly) polynomial-time algorithm [55]. Binary search over λ yields the optimum ratio.

Examples of DSS. We briefly describe several interesting special cases of DSS that have been studied in the literature. Given a graph $G = (V, E)$ we mentioned that $f(S) = |E(S)|$ is supermodular. This also holds when G is a hypergraph and $E(S)$ is the set of hyperedges contained in S [31]. We let r denote the rank of a hypergraph. Let $\mathcal{F}$ be a finite set of pattern graphs; for instance $\mathcal{F}$ can consist of a triangle [64] or k-clique for some k [61]. Given a graph G let $f(S)$ be the number of copies of pattern graphs from $\mathcal{F}$ in the graph $G[S]$. It is not hard to see that f is supermodular for any finite $\mathcal{F}$ [22]; by defining a hyperedge for each copy of a pattern in G we can reduce the pattern graph problem to the hypergraph case where the rank of the resulting hypergraph is the largest size of a pattern graph in $\mathcal{F}$. A further generalization is the class of r-decomposable supermodular functions where f can be written as a non-negative sum of "simpler" supermodular functions, each of which is defined over only a subset of at most r vertices of V (a formal definition is given later). More recently Veldt, Benson, and Kleinberg [65] defined the generalized mean densest subgraph problem (GDSG): for a given real-valued parameter p and an undirected graph G define $f(S) = \sum_{v \in S} d(v, S)^p$ where $d(v, S)$ of degree of v in $G[S]$.[2] When $p = 1$ we obtain DSG, and for any $p \geq 1$ the function is supermodular as observed in [65]. In fact, it can be seen that $\sum_{v \in S} h(d(v, S))$ is supermodular for any real-valued convex function h. [65] is a very recent manuscript indicating the ongoing interest in models and problems related to dense subgraph discovery.

Although DSS is polytime-solvable, the known algorithms for submodular function minimization are slow in theory and practice due to their generality. For the special case of DSG, a polynomial time algorithm is also known via a simple and elegant reduction from the decision problem to finding a maximum flow in an associated directed graph [29, 40, 51]. One can avoid the binary search using parametric maximum flows [26], and similar ideas also apply to DSS. The constructed graph has $O(|V| + |E|)$ nodes and $O(|E|)$ edges. Although maxflow admits faster algorithms, it can take substantial time and space for large scale graphs that arise in several applications. This has motivated the study of heuristics and approximation algorithms.

Peeling Algorithms, Approximation, and Linear Programming. An influential development in fast algorithms for DSG and related problems came from the work of Charikar [14]. He showed that a simple greedy peeling algorithm for DSG that runs in essentially linear time (described by Asahiro et al. [5]) yields a $\frac{1}{2}$-approximation[3]. Second, his analysis was based on an LP formulation whose solution provides an optimum solution for DSG. The peeling algorithm creates an ordering $v_1, v_2, \ldots, v_n$ of the vertices as follows: v_1 is a smallest degree vertex in G, v_2 is a smallest degree vertex in $G - v_1$, and inductively v_i is a smallest degree vertex in $G - \{v_1, \ldots, v_{i-1}\}$. After the ordering is created, the algorithm picks the best density subgraph among all suffixes (of the form $\{v_i, \ldots, v_n\}$) of the ordering. The simplicity and efficiency of the peeling algorithm, and its provable

[1] In this paper, as is standard, we will assume that the set functions are available as a value oracle. We will assume that all supermodular functions considered in this paper are non-negative and normalized ($f(\emptyset) = 0$), which also implies that they are monotone, that is $f(A) \leq f(B)$ for $A \subseteq B$.

[2] [65] consider maximizing the objective $\left(\frac{f(S)}{|S|}\right)^{1/p}$. For $p > 0$, it is equivalent to maximizing $\frac{f(S)}{|S|}$. However, the original objective is needed when $p < 0$, and this regime has interesting applications and connections.

[3] Kortsarz and Peleg observed the $\frac{1}{2}$-approximation earlier in a different context [38] without highlighting peeling.

worst-case guarantee, made it a compelling choice for large scale applications. The algorithm and the analysis have been used in several subsequent papers. The same peeling algorithm creates a core decomposition which was known earlier and is also extensively studied [44]. Charikar's LP relaxation has also been influential since it provided an avenue to use techniques from mathematical programming to solve it exactly or approximately. The dual of the LP turns out to be a mixed packing and covering LP, albeit with additional structure (which can be related to the flow reduction that we mentioned). Bahmani, Goel and Munagala obtained a $(1-\epsilon)$-approximation for DSG in $\tilde{O}(m/\epsilon^2)$ time via the use of the multiplicative weight update (MWU) method, and also devised a parallel algorithm via a clever reformulation of the LP. Here m is the number of edges in the input graph. An important open question in the general context of approximately solving mixed packing and covering LPs is whether one can obtain an ϵ-approximate solution that has near linear dependence on both $1/\epsilon$ and the size of the LP; this is known for pure packing and covering LPs [1, 67] (see [53, 66] for additional results and pointers). Such a result would imply a $(1-\epsilon)$-approximation for DSG in $\tilde{O}(m/\epsilon)$ time. Boob, Sawlani and Wang [13] made partial progress towards this and developed an algorithm for mixed packing and covering that, as a corollary, yields a $(1-\epsilon)$-approximation for DSG in $\tilde{O}(m\Delta/\epsilon)$ time where Δ is the maximum degree. Depending on the range of ϵ relative to Δ, these algorithms are the fastest known $(1-\epsilon)$-approximation algorithms for DSG (in terms of asymptotic running times).

While the peeling algorithm provides a fast and simple algorithm for DSG, the worst case approximation ratio is $\frac{1}{2}$. The approximation ratio becomes larger when considering objectives such as triangle density, k-clique density or other other measures. In a paper that initially inspired our interest, Boob et al. [12] developed a simple *iterative* peeling algorithm for DSG that they called Greedy++. They observed that it is very effective in their experiments and converges to a near-optimal solution in just a few iterations. The algorithm is motivated by an MWU type approach to solve the LP relaxation. The authors conjectured that Greedy++ converges to a $(1-\epsilon)$-approximation in $O(1/\epsilon^2)$ iterations. However, they did not prove anything better than a $\frac{1}{2}$-approximation bound for Greedy++.

1.1 Motivation and Contributions Our discussion of DSG and related problems so far only touches upon a small amount of work on this topic. It should, however, already point to the rich connections to models, algorithms and combinatorial optimization. The impetus for this work came from the following three interrelated high-level questions, with the first one serving as an initial inspiration.

- Does Greedy++ converge to a $(1-\epsilon)$-approximation as the number of iterations grows with ϵ?
- Is there a $(1-\epsilon)$-approximation algorithm for DSG that runs in $\tilde{O}(m/\epsilon)$ time?
- DSG is a special case of DSS which is tractable due to supermodularity. Can we generalize peeling and mathematical programming based approaches in a unified fashion via properties of supermodularity?

In this paper we obtain several results that give affirmative answers to the preceding questions.

Flow strikes back for DSG. Flow-based *exact* algorithms are considered slow for large scale data and, as we remarked, attention has shifted towards approximation algorithms based on peeling and linear programming. In this paper we describe an extremely simple flow-based *approximation* algorithm for DSG. It suffices to obtain an approximate flow in the following sense: the shortest augmenting path length in the residual graph has length $\Omega(\frac{1}{\epsilon}\log m)$. This yields the following theorem which is the first approximation algorithm whose running time has a near-linear dependence on m and $1/\epsilon$.

THEOREM 1.1. *Given an instance of* DSG *on a weighted graph with m edges and n nodes, a $(1-\epsilon)$-approximation can be obtained in $\tilde{O}(m/\epsilon)$ time. For weighted* hypergraphs *the running time is $\tilde{O}(p/\epsilon)$ time where $p = \sum_{e \in E} |e|$ is the total size.*

The algorithm for hypergraphs yields improved running times for various densest pattern subgraph problems; we defer details. We believe that the flow approach can be used to obtain faster approximation algorithms for various other concrete special cases of DSS. The simplicity of the algorithm could lead to new heuristics that can be competitive with other current approaches.

REMARK 1.1. *A $(1-\epsilon)$-approximation to the directed graph version of* DSG *[14, 34, 37, 43] can be obtained via $O(\frac{\log n}{\epsilon})$ calls to a $(1-\epsilon)$-approximation for vertex-weighted* DSG *[54]. Via this reduction we obtain a $(1-\epsilon)$-approximation for directed* DSG *in $\tilde{O}(m/\epsilon^2)$ time.*

Analysis of greedy peeling for DSS. We describe a natural greedy peeling algorithm for DSS and establish an approximation guarantee based on a parameter that depends on f. For a real-valued set function f we consider the parameter $c_f := \max_{S \subseteq V} \frac{\sum_{v \in S} f(v|S-v)}{f(S)}$. It can be seen that $1 \leq c_f \leq |V|$ and $c_f = 1$ iff f is modular[4]. For DSG, $f(S) = |E(S)|$, and it is easy to see that $c_f = 2$ from the observation that sum of degrees is $2|E(S)|$. Similarly for a hypergraph of rank r, $c_f \leq r$.

The peeling algorithm finds $v_1 = \arg\min_{v \in V} f(v \mid V - v)$. It then recursively finds the ordering for $V - v_1$ by restricting f to $V - v_1$, and appends the resulting ordering to v_1. After computing the ordering, it picks, among all suffixes of the ordering, the set S that maximizes the desired ratio $f(S)/|S|$. Via a simple combinatorial analysis (inspired by [3, 35, 37, 38] for DSG), we prove a bound on the approximation ratio achieved by greedy peeling in terms of c_f.

THEOREM 1.2. *Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a nonnegative monotone supermodular function. Then the greedy peeling algorithm for* DSS *on f has an approximation ratio of at least $\frac{1}{c_f}$. For maximizing $f(S)/g(|S|)$ where g is a non-negative concave function, peeling yields an approximation ratio of at least $\frac{1}{(c_f+1)}$.*

We can derive several previous results, in a unified fashion, by upper bounding c_f. We encapsulate two general results in the following and give more details in Section 3. The analysis for these cases is known to be tight from previous work.

COROLLARY 1.1. *Greedy peeling yields an r-approximation for* DSS *when f is an r-decomposable supermodular function. For* GDSG *it yields a $1/(p+1)^p$ approximation.*

REMARK 1.2. *For a supermodular function $f : 2^V \to \mathbb{R}$, we can define a k-core to be a maximal set S such that $f(v \mid S - v) \geq k$ for all $v \in S$. Greedy peeling creates a nested "core" decomposition for any f. Although this property is studied extensively in graphs with several applications [44], the connection to supermodularity is not made explicit — see Theorem 3.2.*

Convergence of iterative peeling algorithms. A formal description of the iterative peeling algorithm Greedy++ suggested in [12] is given in Figure 1 (see Section 4). Recall that Boob et al. conjectured that after $T = O(\frac{1}{\epsilon^2})$ iterations, the algorithm returns a $(1-\epsilon)$-approximate densest subgraph. The first iteration of the algorithm is exactly the greedy peeling algorithm which implies that Greedy++ is a $\frac{1}{2}$-approximation. Despite the very strong form of their conjecture, no better worst-case approximation bound was shown even after an arbitrary number of iterations. We describe a natural extension of Greedy++ to the setting of DSS and we refer to it as Super-Greedy++ (see formal description in Section 4). We show that it converges to a near-optimal solution.

THEOREM 1.3. *Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a nonnegative supermodular function over the ground set V with $n = |V|$.* Super-Greedy++ *outputs a $(1-\epsilon)$-approximate solution after $T = O(\frac{\Delta_f \log n}{\lambda^* \epsilon^2})$ iterations where $\Delta_f = \max_{v \in V} f(V) - f(V - v)$ and λ^* is optimum density. For* DSG, Greedy++ *outputs a $(1-\epsilon)$-approximation after $O(\frac{\Delta \log n}{\lambda^* \epsilon^2})$ iterations where Δ is the maximum degree in the input graph.*

Although we do not resolve the conjecture from [12] in its strongest form, our result captures its spirit, is obtained via a connection to an MWU-based algorithm, and has the right dependence on ϵ. Further, $\frac{\Delta}{\lambda^*}$ is likely to be a small constant in practice. Resolving the precise convergence rate of Greedy++ and Super-Greedy++ in terms of various parameters is an interesting open problem. The key to our proof is to consider the general problem DSS via the Lovász extension of a supermodular function and a corresponding LP relaxation. Our result implies that a simple extension of the peeling algorithm provably converges to an optimum solution for *any* instance of DSS. The impressive practical performance of Greedy++ for DSG, and our result, give an impetus to try iterative peeling for related problems.

The densest at-least-k supermodular subset problem. A useful variant of DSG is the following constrained version: given an additional parameter k, find the densest subgraph among all subgraphs with *at least* k vertices.

[4]The notion of the *curvature*, introduced in [16], has been well-studied in the context of submodular function maximization (see [58]). A similar notion to curvature called *steepness* [32] has been studied in the context of decreasing supermodular functions. Our measure is related but appears to be different in that we consider a ratio that depends on all possible sets rather than on marginal values of individual elements.

Andersen and Chellapilla [3] introduced this problem, and showed that the peeling algorithm with the modification to choose the best suffix with at least k vertices yields a $\frac{1}{3}$-approximation. Khuller and Saha showed that the problem is NP-hard while also giving a $\frac{1}{2}$-approximation [37], and Manurangsi showed that under the small set expansion hypothesis, one cannot obtain a $(\frac{1}{2}+\epsilon)$-approximation for any constant $\epsilon > 0$ [46]. We define the densest at-least-k supermodular subset problem (DalkSS) which is the natural generalization of DSS. We extend the results and algorithms in [3, 37] and prove the following theorem.

Theorem 1.4. *A β-approximation for* DSS *implies a $\frac{\beta}{(\beta+1)}$-approximation for* DalkSS. *In particular there is a $\frac{1}{2}$-approximation for* DalkSS. *The peeling algorithm yields a $\frac{1}{(c_f+1)}$-approximation for* DalkSS.

Along the way we note that the combinatorial algorithm of [37] creates a nested decomposition of dense subgraphs which has been studied as *locally dense* decomposition in [17, 57, 59, 60]. The nested decompositon is very natural in the context of supermodularity under the operation of contraction, while it may appear a bit ad hoc in graphs. We note that our analysis of the peeling algorithm for DalkSS is used in proving the bound for the variant of DSS with a concave denominator (see Theorem 1.2). For the special case of DSG and concave denominator, [35] derived a $\frac{1}{3}$-approximation which follows as a corollary.

There are several other concrete algorithmic consequences, as well as questions, that arise out of our broad investigation in this paper. We defer these to the future.

1.2 Technical Ideas Our first technical contribution is about flow-based approximations for DSG, which produces a fractional b-matching problem. For maximum cardinality bipartite matching, it is well-known that computing an approximate flow such that the shortest augmenting path in the residual network has length at least $1/\epsilon$ yields a $(1 - O(\epsilon))$-approximate matching. This idea has been exploited even in the setting of weighted matchings in general graphs [18], and also for weighted matroid intersection [15]. The density problem is different in that an approximately maximum weight matching does not suffice. A key idea in the proof of Theorem 1.1 is a region-growing argument; similar ideas have appeared previously in local graph clustering [50]. The application of these ideas to DSG appears to be novel.

Our second technical idea is to generalize Charikar's LP for DSG to obtain a mathematical programming relaxation for DSS. For this purpose we use the well-known Lovász extension $\hat{f}$ of a supermodular function f. $\hat{f}$ extends the discrete function to the continuous domain $[0,1]^V$. It is known that $\hat{f}$ is concave whenever f is supermodular. One then obtains a relaxation for DSS as $\max \hat{f}(x)$ subject to $x \in [0,1]^V$ and $\sum_v x_v = 1$. It is not hard to prove that this convex program is an exact relaxation for DSS. A key observation is to rewrite the preceding convex program as a linear program by considering the equivalent objective of minimizing $|S|/f(S)$ which leads to $\min \sum_v x_v$ subject to $x \in [0,1]^V$ and $\hat{f}(x) \geq 1$. The advantage of this reformulation comes via a min-max characterization of $\hat{f}$ that converts it into a large implicit linear program which corresponds to packing *orderings* of the ground set V. This reformulation makes clear the key role that orderings play in DSS and DSG. Our proof of convergence of Super-Greedy++ (and hence Greedy++) is based on interpreting it as a fixed-step size version of MWU to approximately solve this linear program. Although it may appear simple in retrospect, we note that the reformulations are far from clear without the proper general perspective.

Finally, a key to many of our results is to apply the lens of supermodularity. We apply this to peeling algorithms to isolate the dependence of the approximation ratio for DSS, DalkSS and related problems on a single parameter for f. This makes the analysis transparent, unifies past work, and points out the classes of functions for which one can expect good bounds.

1.3 Other Related Work There is extensive work on dense subgraph discovery since it is an important topic in data mining and social network analysis. We already referred to several surveys. Most of the work has been on graph and hypergraph related problems due to their concrete nature and applications. Although supermodularity has been recognized as the reason for tractability of DSG and several related problems, we are not aware of any work that used the lens of supermodularity to unify existing results. In the literature, various objectives of the form $|E(S)|/g(S)$ are considered where g may not be submodular. In such cases the resulting problem often becomes NP-Hard since they capture DkSG or one of its variants; we refer the reader to [62] and [35] where they consider the case when $g(S)$ is a convex function of $|S|$. Recall the GDSG problem in [65]; for $p \geq 1$ it gives rise to a supermodular objective while the setting of $p < 1$ gives rise to non-supermodular functions; their complexity and approximability status is not yet clear.

In the theoretical CS literature there has been extensive work on the densest k-subgraph problem (DkSG) due to the important role it played (and plays) in approximation algorithms. The best known approximation guarantee is $O(\min\{k, n^{1/4+\epsilon}\})$ [9] and there are also lower bounds on various strong mathematical programming relaxations [10]. In contrast, it has been very difficult to prove hardness of approximation. There have been several non-trivial works on hardness (see [2, 24, 36]) before Manurangsi [45] proved almost polynomial-factor hardness under the ETH assumption; proving a similar hardness under $P \neq NP$ remains a challenging open problem. DkSG is also an interesting problem since several other seemingly unrelated problems can be reduced from it. The approximability of DamkSG is closely related to that of DkSG [37].

As we mentioned already, there has been much interest in fast (approximation) algorithms for DSG and related problems. There have been several interesting papers on parallel (PRAM, MapReduce and MPC) and streaming models of computation [7, 8, 11, 21, 27, 47]. One can use sparsification to reduce the graph size as a preprocessing step [21, 47, 48]. Dynamic graph algorithms have been of much interest in both theory and practice and the paper of Sawlani and Wang [54] obtained the first fully dynamic algorithm for $(1-\epsilon)$-approximate DSG with worst-case polylogarithmic update time; the paper has pointers to prior work.

A convex optimization approach to DSG was developed by Danisch et al. [17], and adapted to the k-clique densest subgraph problem [57]. The authors use Charikar's LP relaxation and apply the Frank-Wolfe iterative algorithm and analysis [33], viewing it as a convex program. Each iteration of the algorithm corresponds to a simple procedure and one can use a bound on the iteration count of the Frank-Wolfe algorithm to obtain a convergence guarantee. In fact, as we observe in this paper, one can view Charikar's LP as a convex program via the Lovász extension and apply the same Frank-Wolfe approach to any supermodular function. The convergence guarantee coupled with the time for each iteration yields an overall run time of $\text{poly}(m,n)/\epsilon^2$ for a $(1-\epsilon)$-approximation. Theoretically faster algorithms are known for DSG (and also the hypergraph version) via the methods in [7, 13], however, there is no direct empirical comparison between the various methods.

The directed DSG problem was proposed by Kannan and Vinay [34]. The input is a directed graph $G = (V, E)$ and the goal is to find two sets of vertices S, T (not necessarily disjoint) to maximize the ratio $|E(S,T)|/\sqrt{|S||T|}$ where $E(S,T)$ is the set of arcs with one endpoint in S and the other in T. The directed version generalizes the undirected one. Charikar [14] obtained an exact algorithm for directed DSG based on an LP relaxation as well as a faster 2-approximation. Khuller and Saha [37] obtained a faster flow-based exact algorithm, and also claimed that a peeling based linear-time algorithm was a 2-approximation. Ma et al. [43] showed that the claim in [37] regarding peeling is incorrect, and developed another 2-approximation that runs in $O(m^{2.5})$ time, as well as other results. We mentioned the reduction in [54] (based on some ideas in [14]) that effectively reduces the $(1-\epsilon)$-approximate directed version to the vertex-weighted undirected version.

Organization. In Section 2, we present our flow-based approximation algorithm for DSG. We analyze the greedy peeling algorithm for DSS in Section 3. In Section 4, we show the convergence of Greedy^{++} for DSG by proving the convergence of the natural extension Super-Greedy^{++} for DSS. We present our approximation algorithms for DalkSS in Section 5.

2 Flow-based approximation

We describe a simple and fast $(1-\epsilon)$-approximation for DSG via flow that runs in $\tilde{O}(m/\epsilon)$ time. The algorithm works for hypergraphs with edge and vertex weights. Previous $(1-\epsilon)$-approximation algorithms [7, 12, 13] were based on (non-flow) LP viewpoints, whereas our algorithm demonstrates the efficacy of flow specifically, while also yielding the first algorithm that has near-linear dependence on the input size and $1/\epsilon$.

Let $G = (V, E)$ be a hypergraph with positive edge weights $w : E \to [1, \infty)$. Each hyperedge $e \in E$ (which we simply refer to as an edge) is a subset of vertices and hence $e \subseteq V$. We denote $n = |V|$, $m = |E|$, $p = \sum_{e \in E} |e|$, and $W = \sum_{e \in E} w(e)$. For a set of vertices $U \subseteq V$, $E(U) = \{e \in E \mid e \subseteq U\}$ is the set of edges contained in U, and we let $f(U) \stackrel{\text{def}}{=} \sum_{e \in E(U)} w(e)$ denote the weight of $E(U)$; $f(U)$ is supermodular. The goal is to maximize the ratio $f(U)/|U|$, which is called the *density* of U, over all sets U (where we treat $0/0 = 0$). We also consider the version with vertex weights at the end of the section — see Remark 2.1. The main result of this section is the following approximation algorithm via flow.

Theorem 2.1. *Given a weighted hypergraph of total size p, minimum edge weight 1, and total edge weight W, and parameters $\lambda > 0$ and $\epsilon \in (0,1)$, there is a deterministic algorithm that, in $O\big(p \ln(W) \log\big((m+n)^2/p\big)/\epsilon\big)$*

time[5]*, either:*

1. *Certifies that the densest subgraph of H has density at most λ.*
2. *Returns a subgraph of density at least $(1-\epsilon)\lambda$.*

We can approximate the densest subgraph with an additional $\log n$ factor in the running time.

COROLLARY 2.1. *Given a weighted hypergraph of total size p, rank r, minimum edge weight 1 and total edge weight W, and parameter $\epsilon \in (0,1)$, there is a deterministic algorithm that, in $O\big(p\ln(W)\log\big((m+n)^2/p\big)(\ln(r)+1/\epsilon)\big)$ time, computes a $(1-\epsilon)$-approximate densest subgraph.*

Proof. We obtain an r-approximation to λ^*, the optimum density, via the simple peeling algorithm. Using binary search, in $O(\log r)$ calls to the algorithm in Theorem 2.1 with fixed error tolerance (say $1/2$), we can obtain a 2-approximation to λ^* in total time $O\big(p\ln(W)\log\big((m+n)^2/p\big)\ln r\big)$. For $i \in \mathbb{Z}_{\geq 0}$, let $\epsilon_i = 1/2^i$. We start off with the 2-approximation which is a $(1+\epsilon_0)$-factor approximation. For $i \geq 0$, given an $(1+\epsilon_i)$-approximation to the optimum density, one can compute a $(1+\epsilon_{i+1})$-approximation to the optimum density in $O(1)$ calls to Theorem 2.1 with error parameter ϵ_{i+1}, which takes $O\big(p\ln(W)\log\big((m+n)^2/p\big)/\epsilon_{i+1}\big)$ time. Refine the approximate density in this manner until $\epsilon/2 \leq \epsilon_i \leq \epsilon$, the total running time is dominated by the last iteration, giving $O\big(p\ln(W)\log\big((m+n)^2/p\big)/\epsilon\big)$ time for this second phase. Combining the running time of the two phases yields the desired time bound. ∎

2.1 Flow network and setup We start with a reduction of DSG to flow that generalizes naturally to hypergraphs. The reduction is slightly different from the classical ones [29, 51] and is inspired by the dual of the LP considered by Charikar [14]; as far as we are aware this was first described in [12] . Given a parameter $\lambda > 0$, one can verify that the maximum density is at most λ via flow in the following bipartite flow network $H = (V_H, E_H)$. (We discuss arcs when referring to E_H to distinguish them from undirected edges of G.) V_H consists of the following vertices: a source s, a sink t, a vertex a_e for each edge $e \in E$, and a vertex a_v for every vertex $v \in V$. E_H consists of the following: an arc (s, a_e) with capacity $w(e)$ for every $e \in E$; an arc (a_v, t) with capacity λ for every vertex $v \in V$; and an arc (a_e, a_v) with capacity $+\infty$ for every edge $e \in E$ and every endpoint $v \in e$. The total size of the flow network is $O(p)$. Note that the maximum (s,t)-flow in H is at most W (assuming $W \geq n\lambda$ wlog), as this is the total out-degree of s. We will represent a flow in H by a vector $x \in \mathbb{R}^{E_H}_{\geq 0}$ and let $|x|$ denote the value of flow.

For every set $U \subseteq V$, the vertex set

$$\{s\} \cup \{a_v : v \in U\} \cup \{a_e : e \in E \text{ and } e \subseteq U\}$$

– which gathers all the auxiliary vertices associated with the subgraph $G[U]$, along with s – induces an (s,t)-cut in H of capacity $W - f(U) + \lambda|U|$. In particular, if U has density strictly greater than λ, then the maximum flow has size less than W. (The converse – if the densest subgraph has density at most λ, then the maximum flow has size W – also holds, but here it is not needed.)

Let x be a fixed flow. The arc (s, a_e) is *saturated* if $x(s, a_e) = w(e)$ — and also say that the edge $e \in E$ is saturated — and *unsaturated* otherwise. We let H_x denote the residual graph of x. We call an arc in H_x an *input arc* if it also appears in H, and a *reverse arc* if it does not. For a set of vertices S in H, we let $\partial^+(S \,|\, H_x)$ denote the out-cut of S in the residual graph H_x, and we let $c(\partial^+(S \,|\, H_x))$ denote the total capacity of this cut.

For a fixed flow x, we consider distances from s in H_x w/r/t the number of *reversed* arcs in any path. Let L_i be the ith level set in distance (measured by the number of reverse arcs) from s (but excluding s). We call an arc in H_x a *forwards arc* if it goes from L_i to L_{i+1} for some i; otherwise it is called a *backwards arc.* (Note that "backwards" and "reverse" have different meanings, and that all forwards arcs are reverse arcs.) For each index i, we let $S_i = L_0 \cup L_1 \cup \cdots \cup L_i$ denote the union of the level sets up to L_i. We let $F_i = \{e \in E : a_e \in S_i\}$ denote the set of edges whose auxiliary vertex is in S_i. We let $U_i = \{v \in V : a_v \in S_i\}$ be the set of vertices whose auxiliary vertex is in S_i.

With the flow network and notation as described above, we have the following convenient properties that are easily verified.

[5]For $(1-\epsilon)$-approximations W can be reduced to be a small polynomial in m, n via a standard preprocessing step of removing edges whose weight is much smaller than the heaviest weight edge. Hence, $\ln(W) = O(\ln(m+n))$.

LEMMA 2.1. *For any flow x in H, we have the following.*

1. *If an edge $e \in E$ is unsaturated, then $a_e \in L_0$.*
2. *For all edges $e \in E$, and endpoints $v \in e$, if $a_e \in S_i$, then $a_v \in S_i$.*

2.2 Approximation analysis: long augmenting paths imply dense level sets Through this section, let x be a fixed flow, and let h be the minimum number of reverse arcs along any augmenting path. We analyze the densities of the subgraphs induced by the sets U_i and prove that, unless x already routes W units of flow, larger values of h imply a greater maximum density among these subgraphs.

LEMMA 2.2. *Suppose x is not a maximum flow in H, and the reverse-arc-distance of the shortest augmenting path is at least h. Then there is an index i such that*

$$c\big(\partial^+(s + S_i \,|\, H_x)\big) \leq \Big(W^{1/h} - 1\Big) w(F_i).$$

Proof. Observe that all forwards arcs from L_i to L_{i+1} – i.e., in $\partial^+(s + S_i \,|\, H_x)$ – are of the form (a_v, a_e) for a saturated edge $e \in E$ and an endpoint $v \in e$. For such an edge e, we have $e \in F_{i+1} \setminus F_i$, and the total capacity of its associated forward (and reverse) arcs – (a_{v_i}, a_e) in H_x over $i \in [k]$ – is at most $w(e)$. Charging the capacity of these forward arcs to the weight of e, and summing over all e, we have

$$c\big(\partial^+(s + S_i \,|\, H_x)\big) \leq w(F_{i+1}) - w(F_i) \tag{2.1}$$

for all $i \in \mathbb{Z}_{\geq 0}$.

Now, suppose by contradiction that $c(\partial^+(s + S_i \,|\, H_x)) > \big(W^{1/h} - 1\big) w(F_i)$ for all i. Then for each i, we have

$$w(F_{i+1}) \overset{\text{(a)}}{\geq} c\big(\partial^+(s + S_i \,|\, H_x)\big) + w(F_i) > W^{1/h} w(F_i)$$

where (a) is by (2.1). Unrolling via the above, we have

$$W \geq w(F_h) > W^{1/h} w(F_{h-1}) > \cdots > W w(F_0) \overset{\text{(b)}}{\geq} W,$$

a contradiction. Here (b) is because F_0 is nonempty when x is not a maximum flow (per Lemma 2.1), and every edge has weight at least 1. ∎

LEMMA 2.3. *Suppose x is not a maximum flow in H, and that any augmenting path in H_x has at least h reverse arcs. Then there exists an index i such that*

$$\frac{f(U_i)}{|U_i|} \geq W^{-1/h} \lambda.$$

Proof. Let i be the index asserted by Lemma 2.2. We have

$$\begin{aligned} W &\overset{\text{(c)}}{\geq} |x| = c\big(\partial^+(s + S_i \,|\, H)\big) - c\big(\partial^+(s + S_i \,|\, H_x)\big) \\ &\overset{\text{(d)}}{\geq} \lambda |U_i| + W - f(U_i) - c\big(\partial^+(s + S_i \,|\, H_x)\big) \\ &\overset{\text{(e)}}{\geq} \lambda |U_i| + W - f(U_i) - \Big(W^{1/h} - 1\Big) w(F_i) \\ &\overset{\text{(f)}}{\geq} \lambda |U_i| + W - W^{1/h} f(U_i). \end{aligned}$$

as desired (up to rearrangement). Here (c) is because x is not a max flow. (d) is by construction of H. (e) is by choice of i (per Lemma 2.2). (f) is because $F_i \subseteq E(U_i)$ for all i by Lemma 2.1. ∎

This brings us to the proof of Theorem 2.1.

2.3 Proof of Theorem 2.1 Let $h_0 = \lceil 2\ln(W)/\epsilon \rceil + 2$ and let $h_1 = 2h_0 + 2$. By running blocking flows in H for h_1 iterations, we obtain a flow x where the length of the shortest augmenting path in H_x is at least h_1. Observe that any augmenting path in H_x must alternate between input and reverse edges except at the beginning and end where there are two consecutive input edges. Thus every augmenting path has at least h_0 reverse edges. Either x is a flow of size W – certifying that every subgraph has density at most λ – or by Lemma 2.3, we obtain a set $U \subseteq V$ such that

$$f(U) \geq W^{-1/h_0}\lambda|U| \geq e^{-\epsilon/2}\lambda|U| \geq (1-\epsilon)\lambda|U|,$$

as desired.

As for the running time, each blocking flow takes $O\big(p\log\big((m+n)^2/p\big)\big)$ time [30], so constructing x takes $O\big(p\ln(W)\log\big((m+n)^2/p\big)/\epsilon\big)$ time. It is easy to see that the set U can be obtained in $O(p)$ time.

REMARK 2.1. *Consider the variant where the vertices in the input hypergraph $G = (V, E)$ have non-negative weights given by $g : V \to \mathbb{R}_{>0}$, and the goal is to solve $\max_{S\subseteq V} f(S)/g(S)$. Theorem 2.1 and its corollary easily extend to this setting. The flow network assigns the capacity of the arc (a_v, t) to be $\lambda \cdot g(v)$ instead of λ. The vertex-weighted version is useful in reducing the densest subgraph problem in directed graphs to one in undirected graphs [54].*

3 Greedy peeling algorithm for DSS

Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a nonnegative monotone supermodular function with $f(\emptyset) = 0$. We consider the natural peeling algorithm to find $\max_{S\subseteq V} \frac{f(S)}{|S|}$. Recall from the introduction that the algorithm creates an ordering $v_1, v_2, \ldots, v_n$ of V with the property that for each j, $v_j = \arg\min_{v\in S_j} f(v \mid S_j - v)$ where $S_j = \{v_j, v_{j+1}, \ldots, v_n\}$. The algorithm returns S_j where $j = \arg\max_i \frac{f(S_i)}{|S_i|}$. The analysis is via the parameter c_f. Recall that for any $S \subseteq V$ we have $f(S) \leq \sum_{v\in S} f(v \mid S - v) \leq c_f f(S)$.

THEOREM 3.1. *Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a nonnegative monotone supermodular function where $f(\emptyset) = 0$. The natural greedy peeling algorithm has an approximation ratio of at least $\frac{1}{c_f}$.*

Proof. Let S^* be a maximum density subset and $\lambda^* = \frac{f(S^*)}{|S^*|}$. We claim that for all $v \in S^*$, $f(v \mid S^* - v) \geq \lambda^*$. If S^* is a singleton $\{v\}$, then $f(v \mid S^* - v) = f(S^*) - f(\emptyset) = \lambda^*$. Suppose $|S^*| \geq 2$ and the claim is false. Then $f(S^*) - f(S^* - v) < \frac{f(S^*)}{|S^*|}$. Rearranging, we have $\frac{f(S^*-v)}{|S^*|-1} > \frac{f(S^*)}{|S^*|}$, but then the set $S^* - v$ would contradict the optimality of S^*, thus proving the claim. Let v_j be the first element of S^* in the peeling order. Thus, $v_j \in S_j$ and $S^* \subseteq S_j$ where $f(v_j \mid S_j - v_j) = \min_{u\in S_j} f(u \mid S_j - u)$. From our earlier claim $f(v_j \mid S^* - v_j) \geq \lambda^*$, and by supermodularity $f(v_j \mid S_j - v_j) \geq \lambda^*$ since $S^* \subseteq S_j$. To lower bound the density of S_j, we have

$$\frac{f(S_j)}{|S_j|} = \frac{f(S_j)}{\sum_{u\in S_j} f(u \mid S_j - u)} \cdot \frac{\sum_{u\in S_j} f(u \mid S_j - u)}{|S_j|} \geq \frac{1}{c_f} \cdot \frac{|S_j| f(v_j \mid S_j - v_j)}{|S_j|} \geq \frac{\lambda^*}{c_f}.$$

The first inequality follows from the definition of c_f and the choice of v_j (monotonicity of f implies all marginal values are positive), and the second inequality is from the preceding argument that $f(v_j \mid S_j - v_j) \geq \lambda^*$. S_j is a set considered for output by the algorithm; this concludes the proof. ∎

The preceding proof is similar to the one in [37] for DSG. One can summarize the argument as proving two separate facts: (i) for any j we have $f(S_j)/|S_j| \geq f(v_j \mid S_j - v_j)/c_f$, and (ii) there exists a j such that $f(v_j \mid S_j) \geq \lambda^*$.

Upper bounding c_f. The generic analysis of peeling reduces the approximation analysis to estimating an upper bound on c_f which is often straight forward. We already observed that when $f(S) = |E(S)|$ for a rank r hypergraph, $c_f \leq r$. Rank r hypergraphs implicitly capture various density measures such as triangle and k-clique density [61] and pattern graph density [22]. To further generalize hypergraphs we consider r-decomposable supermodular functions — a similar notion is studied in the submodular minimization literature [6, 20, 49, 56]. We define this class below.

DEFINITION 3.1. *A non-negative supermodular function $f : 2^V \to \mathbb{R}_+$ is r-decomposable if there exist subsets $V_1, V_2, \ldots, V_m$ of V and non-negative supermodular functions $f_1, f_2, \ldots, f_m$ such that (i) $|V_i| \leq r$ for each i (ii) $f_i : 2^{V_i} \to \mathbb{R}_+$ for each i, and (iii) $f(S) = \sum_{i=1}^m f_i(S \cap V_i)$ for each $S \subseteq V$.*

We now upper bound c_f for an r-decomposable supermodular function f.

PROPOSITION 3.1. *For any non-negative r-decomposable supermodular function f, $c_f \leq r$.*

Proof. As f is r-decomposable, for $S \subseteq V$, we can write $f(S) = \sum_{i=1}^{m} f_i(S \cap V_i)$ for nonnegative supermodular functions $f_i : 2^{V_i} \to \mathbb{R}_+$ where $|V_i| \leq r$ for each i. Let $S \subseteq V$. We have

$$\sum_{v \in S} f(v \mid S - v) = \sum_{v \in S} \sum_{i=1}^{m} (f_i(S \cap V_i) - f_i((S \cap V_i) - v)) = \sum_{i=1}^{m} \sum_{v \in S \cap V_i} f_i(v \mid (S \cap V_i) - v).$$

Then

$$\sum_{i=1}^{m} \sum_{v \in S \cap V_i} f_i(v \mid (S \cap V_i) - v) \leq \sum_{i=1}^{m} |V_i| \cdot f_i(S \cap V_i) \leq r \cdot \sum_{i=1}^{m} f_i(S \cap V_i) = r \cdot f(S),$$

where the first inequality follows by the nonnegativity of the f_i. This implies $c_f \leq r$. ∎

Another interesting class of supermodular functions associated with graphs was considered recently in [65]. Given a graph $G = (V, E)$ and a parameter p they consider the function $g_p(S) = \sum_{v \in S} d_S(v)^p$ where $d_S(v)$ is the degree of v in the induced graph $G[S]$. They observed that for $p \geq 1$, g_p is supermodular. They prove that peeling with respect to g_p gives a $1/(p+1)$ approximation, yielding a $1/(p+1)^{1/p}$ approximation for the GDSG problem. Implicit in their proof is the following claim that we reprove formally for the sake of completeness.

PROPOSITION 3.2. *For the function $f = g_p$ when $p \geq 1$, $c_f \leq p + 1$.*

Proof. Let $N(v) = \{u \in V : uv \in E\}$ denote the neighborhood of v in G. As the function $h(x) = x^p$ is convex in x, $h(y) \geq h(z) + h'(z)(y - z)$ for all $y, z \in \mathbb{R}_{\geq 0}$ where h' denotes the derivative of h. Setting $z = x$ and $y = x - 1$, we have $x^p - (x-1)^p \leq px^{p-1}$. Then

$$\begin{aligned}
\sum_{v \in S} g_p(v \mid S - v) &= \sum_{v \in S} \left(d_S(v)^p + \sum_{u \in N(v) \cap S} d_S(u)^p - (d_S(u) - 1)^p \right) \\
&\leq \sum_{v \in S} \left(d_S(v)^p + \sum_{u \in N(v) \cap S} p \cdot d_S(u)^{p-1} \right) \\
&= (p+1) \sum_{v \in S} d_S(v)^p \\
&= (p+1) \cdot g_p(S),
\end{aligned}$$

implying $c_f \leq p + 1$. ∎

Density with concave denominator. Consider the problem of finding a set S minimizing $\frac{f(S)}{g(|S|)}$ where g is a real-valued non-negative concave function. This problem is polynomial-time solvable since g is a submodular function. Here we consider the approximation provided by the peeling algorithm. Let $v_1, v_2, \ldots, v_n$ be the order produced by the greedy peeling algorithm for the DSS on input f and let $S_j = \{v_j, v_{j+1}, \ldots, v_n\}$ for $j \in [n]$. We show that the algorithm that returns the set S_j maximizing $\frac{f(S_j)}{g(|S_j|)}$ yields a $\frac{1}{c_f+1}$-approximation. The problem and analysis is inspired by the work of Kawase and Miyauchi who considered maximizing $\frac{|E_G(S)|}{g(|S|)}$ for an undirected graph G [35]. We leverage the analysis of the greedy peeling algorithm for DALKSS, and for this reason we describe it in Section 5.3 after we consider DALKSS.

Core decomposition of a supermodular function. It is well-known that greedy peeling yields a "core" decomposition for an undirected graph. In fact this is a property of supermodularity. To understand this we first consider the definition of a *k-core* of a graph $G = (V, E)$: a k-core is a *maximal* subset $S \subseteq V$ of vertices such that the degree of every vertex in $G[S]$ is at least k. It is not hard to establish that the k-core is unique for any given k

Greedy++$(G = (V, E), T \in \mathbb{Z}_+)$
  for all $v \in V$, set $\ell_v^{(0)} = 0$
  for i from 1 to T **do**
    $S_{i,1} \leftarrow V$
    for j from 1 to $n-1$ **do**
      $v_{i,j} \leftarrow \arg\min_{v \in S_{i,j}} \ell_v^{(i-1)} + \deg_{G[S_{i,j}]}(v)$
      $\ell_{v_{i,j}}^{(i)} \leftarrow \ell_{v_{i,j}}^{(i-1)} + \deg_{G[S_{i,j}]}(v)$
      $S_{i,j+1} \leftarrow S_{i,j} - v_{i,j}$
    end for
  end for
  return $\arg\max_{S_{i,j}} \frac{|E(S_{i,j})|}{|S_{i,j}|}$

Super-Greedy++$(f : 2^V \to \mathbb{R}_{\geq 0}, T \in \mathbb{Z}_+)$
  for all $v \in V$, set $\ell_v^{(0)} = 0$
  for t from 1 to T **do**
    $S_{t,1} \leftarrow V$
    for j from 1 to n **do**
      $v_{t,j} \leftarrow \arg\min_{v \in S_{t,j}} \ell_v^{(t-1)} + f(v \mid S_{t,j} - v)$
      $\ell_{v_{t,j}}^{(t)} \leftarrow \ell_{v_{t,j}}^{(t-1)} + f(v_{t,j} \mid S_{t,j} - v_{t,j})$
      $S_{t,j+1} \leftarrow S_{t,j} - v_{t,j}$
    end for
  end for
  return $\arg\max_{S_{t,j}} \frac{f(S_{t,j})}{|S_{t,j}|}$

Figure 1: The Greedy++ algorithm of Boob et al. [12] and an extension to DSS called Super-Greedy++.

and that if S_1 is the k_1-core and S_2 is the k_2-core with $k_1 \leq k_2$ then $S_2 \subseteq S_1$. In other words the k-cores form a nested sequence. In fact, the peeling algorithm generates all the nested k-cores as suffixes of the ordering it produces. We refer the reader to [44] for an extensive survey. For a supermodular function $f : 2^V \to \mathbb{R}_+$ we define the k-core as a maximal subset $S \subseteq V$ such that $f(v \mid S - v) \geq k$ for all $v \in S$. It is not hard to verify that greedy peeling creates a core decomposition for any supermodular function. We believe that this property is known to experts but does not seem to be explicit in the literature. For this reason we capture the property in a theorem and provide a short proof.

THEOREM 3.2. *Let $f : 2^V \to \mathbb{R}_+$ be a supermodular function. Let $v_1, v_2, \ldots, v_n$ be the ordering produced by the greedy peeling algorithm on f. For any number γ there is a unique γ-core $C(\gamma)$ of f, and moreover $C(\gamma) = S_i$ where i is the smallest index such that $f(v_i \mid S_i - v_i) \geq \gamma$.*

Proof. Suppose A and B are both γ-cores and $A \neq B$. For any $v \in A \cup B$ we have $f(v \mid A - v) \geq \gamma$ and $f(v \mid B - v) \geq \gamma$ since A, B are γ-cores, and this implies, by supermodularity that $f(v \mid A \cup B - v) \geq \gamma$. This contradicts maximality of A, B and hence there is a unique γ-core.

Let $C(\gamma)$ be the γ-core and let v_j be the first element of $C(\gamma)$ in the peeling order. This implies that $C(\gamma) \subseteq S_j$. Since $v_j \in C(\gamma)$, $f(v_j \mid C(\gamma) - v_j) \geq \gamma$, and by supermodularity, $f(v_j \mid S_j - v_j) \geq \gamma$ because $C(\gamma) \subseteq S_j$. From the peeling property it follows that for all $u \in S_j$, $f(u \mid S_j - v_j) \geq \gamma$. Since $C(\gamma)$ is unique we must have $S_j = C(\gamma)$. ∎

4 Iterative Peeling for DSG and DSS

As discussed in the introduction, this section is motivated by the work of [12] who proposed the algorithm Greedy++ for DSG. They conjectured that Greedy++ converges to a $(1-\epsilon)$-approximation in $O(1/\epsilon^2)$ iterations. In this section, we present and analyze an algorithm for the more general DSS problem called Super-Greedy++. See Figure 1 for a formal description.

We first give an informal description of Super-Greedy++ and offer some intuition (which comes from [12] via Greedy++). Recall that a single iteration of the peeling algorithm already gives an approximation for DSS. Super-Greedy++, following Greedy++, tries to improve the approximation by iteratively running the peeling algorithm, with a similar greedy peeling process that are now based on *loads*. Loads, which are defined element-wise, are inspired by the load-balancing LP for DSG, and incorporate information from earlier iterations of peeling. The load of an element v is denoted ℓ_v, and initially all vertices v have load 0. In each iteration, the load-based peeling process peels off the element v that minimizes $\ell_v + f(v \mid S - v)$, where S denotes the set of vertices remaining at an intermediate point in the peeling process. For the peeled off element v, we increase ℓ_v by $f(v \mid S - v)$. Observe that the first iteration, where $\ell_v^{(0)} = 0$, coincides with the standard peeling algorithm analyzed in Section 3. Selecting v to minimize $\ell_v + f(v \mid S - v)$ can be interpreted as a greedy heuristic to minimize the maximum load over all the

elements (partly because by supermodularity, the increase of $f(v \mid S - v)$ dwindles as elements are peeled from S). This reflects the intuition behind the standard greedy peeling algorithm, except now in a certain aggregate sense. The hope is that the loads converge to an optimum solution to the load-balancing LP (for DSG). For each iteration of load-based peeling, the suffixes of the peeling order are candidates for the densest supermodular set. Super-Greedy++ outputs the densest suffix over all of its iterations.

Super-Greedy++ is appealing because it is simple and intuitive. We prove that it also converges to optimum solutions to DSS, as follows.

THEOREM 4.1. *Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a normalized, nonnegative, and monotone supermodular function. Let $\epsilon \in (0, 1)$. Let $\Delta = \max_{v \in V} f(v \mid V)$ and let $\lambda^\star$ be the maximum density. For $T \geq O\left(\frac{\Delta \ln(n)}{\lambda^\star \epsilon^2}\right)$, Super-Greedy++ outputs a $(1-\epsilon)$-approximation to* DSS.

Therefore, as a special case of Super-Greedy++, Greedy++ outputs a $(1-\epsilon)$-approximation in $O(\frac{\Delta \ln(n)}{\lambda^\star \epsilon^2})$ iterations where Δ is the maximum degree of the graph.

The remainder of this section is dedicated to proving Theorem 4.1. Throughout this section, we let f, Δ, $\lambda^\star$, T, and ϵ be fixed and as described in Theorem 4.1. For ease of exposition, we will instead prove Theorem 4.1 with a $(1 - c\epsilon)$-approximation for some constant $c \geq 1$, rather than $1 - \epsilon$. The $(1-\epsilon)$-approximation bound then follows from decreasing ϵ by a constant factor, which only increases the constant hidden in the $O\left(\frac{\Delta \ln(n)}{\lambda^\star \epsilon^2}\right)$.

The high-level argument is as follows. First, we identify a different packing LP for DSS, where solutions to the dual covering LP can be rounded to solutions to DSS. We then describe a robust MWU algorithm for this packing LP (which also gives solutions to the dual) and describe standard bounds that match Theorem 4.1 (but for a different algorithm). Having established the MWU framework as an algorithmic template, we argue that Super-Greedy++ is in fact a special case of the MWU framework, and that it is also implicitly rounding the dual solution provided by MWU (approximately) on-the-fly. In the end, we obtain the bounds of Theorem 4.1 by proxy to the MWU framework, for a carefully selected packing LP formulation of DSS.

4.1 The Lovász extension, contrapolymatroids, and packing and covering LPs based on orderings

As discussed, there are multiple ways to express DSG as a linear program. The flow formulation of [29, 51], the load balancing LP via the dual of the LP of Charikar [14] (motivating [12, 13]), the low-width packing formulation of [7], and the associated dual LPs all model DSG yet lead to very different algorithms and bounds. However none of these LP's fully explain Greedy++. Another shortcoming of existing LP's is the explicit dependence on the underlying graph. To generalize to DSS, we require an LP that abstracts out the underlying graph. In this section, with these motivations in hand, we derive another LP for DSS. Our approach is based on connections between the Lovász extension, contrapolymatroids, and the symmetric group of V, building on classical work by Edmonds [19] and Lovász [42].

Let $\hat{f} : [0,1]^E \to \mathbb{R}_{\geq 0}$ denote the *Lovász extension* of f. There are multiple equivalent definitions of $\hat{f}$ (cf. [42]) and here we give a concrete one convenient for our discussion. Given $x \in [0,1]^E$, and $\tau \in [0,1]$, let $S_\tau = \{v \in V : x_v \geq \tau\}$. Then

$$\hat{f}(x) = \mathop{\mathbf{E}}_{\tau}[f(S_\tau)]$$

where the randomness is over $\tau \in [0,1]$ drawn uniformly at random. It is well-known that $\hat{f}$ is a (piece-wise linear) concave function iff f is supermodular (equivalently it is convex iff f is submodular). Consider now the following maximization problem.

$$\text{maximize } \hat{f}(x) \text{ over } x \in \mathbb{R}^V_{\geq 0} \text{ s.t. } \sum_{v \in V} x_v \leq 1. \tag{4.2}$$

(4.2) is essentially equivalent to DSS in the following sense.

LEMMA 4.1. (4.2) *is an exact formulation for* DSS. *Moreover:*

1. *Given a set S with density λ, the vector $x = \mathbb{1}_S/|S|$ is feasible in* (4.2) *and has value $\hat{f}(x) = \lambda$.*
2. *Given a feasible solution x to* (4.2)*, there exists a value $\tau \in [0,1]$ such that S_τ has density at least $\hat{f}(x)$.*

Proof. The initial claim regarding the objective value follows from the two subsequent items. The first item follows directly from the definition of $\hat{f}(x)$. For the second item, fix a feasible x for (4.2), and suppose by contradiction that all sets of the form S_τ have density less than $\hat{f}(x)$. Then

$$\hat{f}(x) = \mathbf{E}[f(S_\tau)] < \hat{f}(x)\,\mathbf{E}[|S_\tau|] = \hat{f}(x)\langle \mathbb{1}, x\rangle = \hat{f}(x),$$

a contradiction. ∎

Charikar's LP for DSG [14] is a special case of (4.2). This can be seen by noting that $\hat{f}(x) = \sum_{uv\in E} \min\{x_u, x_v\}$ when $f(S) = |E(S)|$ for a given graph $G = (V, E)$; one can then rewrite the concave objective as a linear objective via additional variables y_e for $e \in E$. In general, (4.2) is not a linear program. Towards an LP formulation, let S_V denote the symmetric group (i.e., the set of all permutations, or equivalently, the set of all orderings) of V. For an ordering $\sigma \in S_V$, consider the vector $q(\sigma) \in \mathbb{R}^V_{\geq 0}$ defined by

$$q_v(\sigma) = f(v \,|\, \{w : w \prec_\sigma v\})$$

for $v \in V$. Then $\hat{f}$ can now be described alternatively as follows.

LEMMA 4.2. *For all $x \in [0,1]^V$, $\hat{f}(x) = \min_\sigma \langle x, q(\sigma)\rangle$. Moreover, for a given x, $\min_\sigma \langle x, q(\sigma)\rangle$ is attained by the ordering σ_x that sorts the elements of V in* decreasing *order of values x_v.*

The proof of the first claim Lemma 4.2 follows from combining work of Edmonds [19] (who showed that the points $q(\sigma)$ give the vertices of the base contrapolymatroid of f) and Lovász [42] (who showed that $\hat{f}(x)$ minimizes $\langle x, p\rangle$ over all points p in the base contrapolymatroid). The second claim comes from the greedy algorithm of [19]. (See also [55].)

The characterization of the Lovász extension given in Lemma 4.2 allows us to express (4.2) as an LP. Recall that $\lambda^\star$ denotes the optimum value of DSS and (4.2), and consider the following LP.

$$\text{minimize } \sum_{v\in V} x_v \text{ over } x \geq \mathbb{0} \text{ s.t. } \langle x, q(\sigma)\rangle \geq \lambda^\star \text{ for all } \sigma \in S_V. \tag{4.3}$$

The covering constraints over $\sigma \in S_v$ are equivalent to the requirement that $\hat{f}(x) \geq \lambda^\star$. Clearly, by choice of $\lambda^\star$, the optimum value of this covering LP is 1, and any optimum solution of (4.3) is also an optimum solution to (4.2). The dual of (4.3) is the following packing LP:

$$\text{maximize } \lambda^\star \sum_{\sigma\in S_V} y_\sigma \text{ over } y : S_V \to \mathbb{R}_{\geq 0} \text{ s.t. } \sum_{\sigma\in S_V} y_\sigma q_v(\sigma) \leq 1 \text{ for all } v \in V. \tag{4.4}$$

This final LP – packing (vertices of the contrapolymatroid of f induced by) orderings, into $\mathbb{1}^V$ – is the LP that we will argue that Super-Greedy++ is implicitly solving. We will also argue Super-Greedy++ is solving (and rounding) the dual covering LP (4.3). Before entering this next segment of our proof, let us summarize our discussion in the following lemma. We omit the proof which is straightforward given the description above.

LEMMA 4.3. (4.3) *and* (4.4) *have optimum value* 1. *Given a feasible solution x to* (4.3)*, there exists $\tau \in [0,1]$ such that the set $S_\tau = \{v \in V : x_v \geq \tau\}$ has density $\lambda^\star/\langle \mathbb{1}, x\rangle$.*

4.2 Solving packing LP (4.4) **via MWU** In the preceding section, we identified dual pure packing and covering LP's, (4.4) and (4.3), that model DSS. Observe that the packing LP (4.4) has n constraints indexed by V and exponentially many variables indexed by S_V.

For a set of weights $w : V \to \mathbb{R}_{\geq 0}$, the Lagrangian relaxation of (4.4) induced by w (collapsing the n packing constraints to a single one) is solved by identifying the ordering $\sigma \in S_V$ minimizing $\langle w, q(\sigma)\rangle$. As noted in Lemma 4.2, this is polynomial time solvable by taking $\sigma \in S_V$ to be the ordering that is decreasing in w_v. This feature makes (4.4) well-suited to the MWU framework which approximates (4.4) via a sequence of approximate solutions to such relaxations.

Order-Packing-MWU($f : 2^V \to \mathbb{R}_{\geq 0}, \epsilon \in (0,1), T \in \mathbb{N}, \alpha \geq 1$)
  $w^{(0)} \leftarrow \mathbb{1}_V$, $y^{(0)} \leftarrow \mathbb{0}_{S_V}$, and $\eta \leftarrow \frac{\ln n}{\epsilon}$
  for t from 1 to T **do**
    $\sigma^{(t)} \leftarrow$ any $\sigma \in S_V$ such that
    $\langle w^{(t)}, q(\sigma)\rangle \leq \alpha \min_{\sigma' \in S_V} \langle w^{(t)}, q(\sigma')\rangle$
    $y^{(t)} \leftarrow y^{(t-1)} + \frac{1}{\lambda^\star T} 1_{\sigma^{(t)}}$
    for $v \in V$ **do**
      $w_v^{(t+1)} \leftarrow w_v^{(t)} \cdot e^{\frac{\eta}{\lambda^\star T} q_v(\sigma^{(t)})} = e^{\eta \sum_\sigma y_\sigma^{(t)} q_v(\sigma)}$
    end for
  end for
  return $y^{(T)}$ // $= \frac{1}{\lambda^\star T} \sum_{t=1}^T 1_{\sigma^{(t)}}$

Figure 2: The MWU framework applied to the packing LP (4.4).

When we apply the MWU framework to (4.4), with a fixed (i.e., width-dependent) step size[6], we obtain the algorithm Order-Packing-MWU described in figure Figure 2[7]. We highlight some salient features of this algorithm. First, the parameter Δ plays a role as it bounds the maximum load induced on a single packing constraint by any choice of σ. Meanwhile the optimum value of the LP is $1/\lambda^\star$ which constrains the sum of y_σ's. This leads to an iteration count proportional to $\Delta/\lambda^\star$. Second, the MWU framework is robust to approximate solutions to the Lagrangian relaxation, where an α-approximation to the relaxation leads to an additional α-multiplicative factor in the approximation of the ultimate solution. This tolerance to local approximation plays a critical role in the overall argument, as we will later prove that the iterative peeling step in Super-Greedy++ produces orderings $\pi^{(t)}$ that approximate these Lagrangian relaxations. (We have encoded α explicitly as a parameter to Order-Packing-MWU.) Standard analyses for the MWU framework implies the following bounds.

LEMMA 4.4. *$T \geq O\big(\Delta \ln(n)/\lambda^\star \epsilon^2\big)$, and $\alpha \geq 1$, Order-Packing-MWU has the following guarantees, where we let $\beta = (1+\epsilon)\alpha$.*

1. *The vector $\beta^{-1} y^{(T)}$ is a β^{-1}-approximate solution to the packing LP (4.4).*
2. *For some iteration $t \in [T]$, the point $\beta w^{(t)}/\langle \mathbb{1}, w^{(t)}\rangle$ is a feasible β-approximate solution for the dual covering LP (4.3).*

Proof. The proof is standard and included for the sake of completeness[8]. For ease of readability, we prove the lemma except with the $(1+\epsilon)$ replaced with $(1+2\epsilon)$. The claimed bounds then follow from replacing ϵ with $\epsilon/2$ by a constant, which only increases the hidden constant in T.

Let $\beta = (1+2\epsilon)\alpha$. We first address the first claim, that $\beta^{-1} y^{(T)}$ is a β^{-1}-approximation to (4.4). Clearly $y^{(T)}$ has objective value $\lambda^\star \langle \mathbb{1}, y^{(T)}\rangle = \lambda^\star \sum_{t=1}^T \frac{1}{\lambda^\star T} = 1$, which is the optimum value of (4.4). We claim that $y^{(T)}$ violates the packing constraints by at most a β-multiplicative factor. If so, then scaling $y^{(T)}$ down by β makes it a feasible β^{-1}-approximation.

Let $y^\star : S_V \to \mathbb{R}_{\geq 0}$ be an optimal solution to (4.3). In particular, we have $\sum_\sigma y_\sigma^\star = 1/\lambda^\star$ and $\sum_\sigma y_\sigma^\star q_v(\sigma) \leq 1$ for all $v \in V$. Now, for each iteration t, we have

$$\langle w^{(t)}, q(\sigma^{(t)})\rangle \leq \frac{\alpha}{\sum_\sigma y_\sigma^\star} \sum_\sigma y_\sigma^\star \langle w^{(t)}, q(\sigma)\rangle \leq \frac{\alpha}{\sum_\sigma y_\sigma^\star} \langle w^{(t)}, \mathbb{1}\rangle = \alpha \lambda^\star \langle w^{(t)}, \mathbb{1}\rangle. \tag{4.5}$$

[6] MWU is an extensively used technique for solving linear programs, and there are several variants [4]. The one that we need is closest to the algorithm from [52]. We give a short self-contained analysis using our terminology.

[7] The algorithm Order-Packing-MWU assumes access to $\lambda^\star$, which is not known *a priori*. This is valid because here Order-Packing-MWU serves only as an idealized algorithm that facilitates the analysis of Super-Greedy++. One can rewrite Order-Packing-MWU to avoid the dependence on $\lambda^\star$; however, the relation to Super-Greedy++ becomes more complicated.

[8] Particularly as there are some competing notions of "width" in the literature for DSG.

Consequently, for each iteration t, we have

$$\langle \mathbb{1}, w^{(t+1)} \rangle = \sum_v w_v^{(t)} e^{\frac{\eta}{\lambda^\star T} q_v(\sigma^{(t)})} \overset{\text{(g)}}{\leq} \langle \mathbb{1}, w^{(t)} \rangle + \frac{(1+\epsilon)\eta}{\lambda^\star T} \langle w^{(t)}, q(\sigma^{(t)}) \rangle$$
$$\leq 1 + \frac{(1+\epsilon)\alpha\eta}{T} \langle \mathbb{1}, w^{(t)} \rangle \leq e^{(1+\epsilon)\alpha\eta/T} \langle \mathbb{1}, w^{(t)} \rangle.$$

Here (g) is by the inequality $e^x \leq 1 + x + x^2$ for small x, noting that $\frac{\eta}{\lambda^\star T} q(\sigma_v^{(t)}) \leq \epsilon q(\sigma_v^{(t)})/\Delta \leq \epsilon$. Unrolling the inequality obtained above from $w^{(T)}$ down to $w^{(0)} = \mathbb{1}$ gives $\langle \mathbb{1}, w^{(T)} \rangle \leq e^{\epsilon\alpha\eta} n$. Now, for each element v, we have

$$\sum_\sigma y_\sigma^{(T)} q_v(\sigma) = \frac{1}{\eta} \ln(w^{(T)}) \leq \frac{1}{\eta} \ln(\langle \mathbb{1}, w^{(T)} \rangle) \leq (1+\epsilon)\alpha + \epsilon \leq \beta,$$

as desired.

Suppose by contradiction that for all iterations t, $x^{(t)} = \beta w^{(t)} / \langle \mathbb{1}, w^{(t)} \rangle$ was not a β-approximation. In particular, $x^{(t)}$ is not feasible for (4.3). Then for every iteration t, instead of (4.5), we obtain the stronger inequality,

$$\langle w^{(t)}, q(\sigma^{(t)}) \rangle = \frac{\langle \mathbb{1}, w^{(t)} \rangle}{\beta} \langle x^{(t)}, q(\sigma^{(t)}) \rangle \leq \frac{\alpha \langle \mathbb{1}, w^{(t)} \rangle}{\beta} \min_{\sigma'} \langle x^{(t)}, q(\sigma') \rangle < \frac{\alpha \langle \mathbb{1}, w^{(t)} \rangle}{\beta} \lambda^\star. \tag{4.6}$$

(4.6) is stronger than the inequality (4.5) by a factor of β (and makes the inequality strict). Then repeating the same calculations as for the first claim above, except using (4.6) instead of (4.5), improves the bounds by a factor of β and leads us to conclude $\sum_\sigma y_\sigma^{(T)} q_v(\sigma) < 1$. But then scaling up $y^{(T)}$ slightly gives a feasible solution with objective value better than $\lambda^\star$, a contradiction. This establishes the second claim and completes the proof. ∎

4.3 Super-Greedy++ as a special case of MWU Each iteration of Super-Greedy++ peels the elements in some order. Let $\pi^{(t)} \in S_V$ denote the *reverse* of this peeled ordering in iteration t. In this section, we recast Super-Greedy++ as a special case of Order-Packing-MWU, where the reverse-peeled orderings $\pi^{(t)}$ assume the roles of orderings $\sigma^{(t)}$ selected in Order-Packing-MWU.

LEMMA 4.5. Super-Greedy++ *is an instance of Order-Packing-MWU with* $\alpha = e^\epsilon$ *via the following correspondence for each iteration* $t \in [T]$.

1. *For each* $v \in V$, $\ell_v^{(t)}$ *in* Super-Greedy++ *corresponds to* $\lambda^\star T \sum_\sigma y_\sigma^{(t)} q_v(\sigma)$ *in Order-Packing-MWU, thereby* $w^{(t)} = e^{(\eta/\lambda^\star T)\ell_v^{(t)}}$.
2. *Each* $\pi^{(t)}$ *in* Super-Greedy++ *corresponds to* $\sigma^{(t)}$ *in Order-Packing-MWU.*

Proof. The claim in item 1 holds initially for $t = 0$ when $\ell^{(0)} = \mathbb{0}$ and $w^{(0)} = \mathbb{1}$. The key claim is that

Each $\pi^{(t)}$ *is an* e^ϵ*-approximation to the ordering* $\sigma \in S_V$ *minimizing* $\langle w^{(t)}, q(\sigma) \rangle$.

Assuming this claim, it is easy to see that the subsequent increase in $\ell^{(t)}$ in Super-Greedy++ matches the subsequent increase in $w^{(t)}$ to maintain the correspondence in item 1. Here we point out that since $\pi^{(t)}$ is the reverse of the peeling order, $q_{v_{t,j}}(\pi^{(t)}) = f(v_{t,j} \mid S_{t,j} - v_{t,j})$ for all j. We then have $\sum_\sigma y_\sigma^{(t)} q_v(\sigma) = \frac{1}{\lambda^\star T} \sum_t \sigma^{(t)} q_v(\sigma^{(t)}) = \frac{1}{\lambda^\star T} \ell_v^{(t)}$, and $w^{(t)} = \exp(\eta \sum_\sigma y_\sigma^{(t)} q_v(\sigma)) = \exp\big(\frac{\eta}{\lambda^\star T} \ell_v^{(t)}\big)$, for all t.

To prove the claim, fix an iteration t, and let $\gamma^{(t)}$ be the ordering decreasing in $w_v^{(t)}$. By Lemma 4.2, $\gamma^{(t)}$ minimizes $\langle w^{(t)}, q(\sigma) \rangle$ over all orderings $\sigma \in S_V$. We prove in Lemma 4.6 below that there exists a vector $\tilde{w}^{(t)} : V \to \mathbb{R}_{>0}$ such that (a) $w^{(t)} \leq \tilde{w}^{(t)} \leq e^\epsilon w^{(t)}$ and (b) $\pi^{(t)}$ is decreasing in $\tilde{w}^{(t)}$. Assuming such a vector $\tilde{w}^{(t)}$, we then have

$$\langle q(\pi^{(t)}), w^{(t)} \rangle \overset{\text{(h)}}{\leq} \langle q(\pi^{(t)}), \tilde{w}^{(t)} \rangle \overset{\text{(i)}}{\leq} \langle q(\gamma^{(t)}), \tilde{w}^{(t)} \rangle \overset{\text{(j)}}{\leq} e^\epsilon \langle q(\gamma^{(t)}), w^{(t)} \rangle,$$

as desired. Here (h) is because $q(\pi^{(t)})$ is nonnegative by monotonicity of f, and $w^{(t)} \leq \tilde{w}^{(t)}$. (i) is because, by Lemma 4.2, $\pi^{(t)}$ minimizes $\langle q(\sigma), \tilde{w}^{(t)} \rangle$ over all orderings σ. (j) is because $q(\gamma^{(t)})$ is nonnegative and $\tilde{w}^{(t)} \leq e^\epsilon w^{(t)}$. ∎

It remains to prove Lemma 4.6, as follows.

LEMMA 4.6. *For all iterations $t \in [T]$, there exists a vector $\tilde{w}^{(t)} \in \mathbb{R}^V_{>0}$ such that (a) $w^{(t)} \le \tilde{w}^{(t)} \le e^{\epsilon} w^{(t)}$ and (b) $\pi^{(t)}$ is decreasing in $\tilde{w}^{(t)}$.*

Proof. Fix t. For ease of notation, we omit t and denote $w = w^{(t)}$, $\ell = \ell^{(t)}$, and $\pi = \pi^{(t)}$. Let γ be the ordering decreasing in ℓ_v, which is also the ordering decreasing in w_v because w_v exponentiates ℓ_v. Define a vector $\tilde{w} \in \mathbb{R}^V_{>0}$ (which will be the desired vector $\tilde{w}^{(t)}$) by setting

$$\tilde{w}_v = \max_{u \succeq_\pi v} w_u$$

for all elements v. Clearly π is decreasing in $\tilde{w}_v$, and $w \le \tilde{w}$. It remains to show that $\tilde{w} \le e^{\epsilon} w$.

Fix $v \in V$. If $\tilde{w}_v > w_v$, then $\tilde{w}_v = w_u$ for an element u with $v \prec_\pi u$. We have $\ell_u \le \ell_v + \Delta$ because otherwise v would be peeled before u, hence $u \prec_\pi v$, a contradiction. In terms of weights, we have

$$\tilde{w}_v = w_u = e^{(\eta/\lambda^\star T)\ell_u} \le e^{(\eta/\lambda^\star T)\ell_v + (\eta\Delta/\lambda^\star T)} \le e^{(\eta/\lambda^\star T)\ell_v + \epsilon} = e^{\epsilon} w_v,$$

as desired. ∎

4.4 Completing the proof of Theorem 4.1 The previous section helps us understand Super-Greedy++ within the MWU framework. We now leverage this viewpoint to argue that the set output by Super-Greedy++ is indeed a good approximation of the densest subset, completing the proof of Theorem 4.1.

Consider again Order-Packing-MWU for $\alpha = e^{\epsilon} = (1 + O(\epsilon))$. Recall from Lemma 4.4 that for some iteration t, the point $x^{(t)} = (1 + O(\epsilon))w^{(t)}/\langle \mathbb{1}, w^{(t)} \rangle$ induces a $(1 + O(\epsilon))$-approximate solution to the dual covering LP (4.3). Then, by Lemma 4.3, there is a set of the form $S_\tau = \{v : x^{(t)}(v) \ge \tau\}$ that has density $(1 - O(\epsilon))\lambda^\star$. One way to generate this set is to index $V = \{v_1, \ldots, v_n\}$ in decreasing order of $x^{(t)}$; then some prefix of the form $S_i = \{v_1, \ldots, v_i\}$ will be the desired set. Note that $x^{(t)}$ and $w^{(t)}$ induce the same ordering on V, so one can instead generate the sets S_i based on $w^{(t)}$.

We argue that this is what Super-Greedy++ is doing, however approximately and indirectly. We first give some intuition. Super-Greedy++ does not maintain the weights $w^{(t)}$ explicitly and the reverse-peeled ordering $\pi^{(t)}$ is not necessarily decreasing in $w^{(t)}$. However we now know that $\pi^{(t)}$ is decreasing w/r/t a vector $\tilde{w}^{(t)}$ that closely approximates $w^{(t)}$. The sets $S_{t,j}$ considered by Super-Greedy++ are implicitly rounding $\tilde{w}^{(t)}$, and thereby approximately rounding $w^{(t)}$.

We now make this argument precise. Let t be an iteration where the point $x^{(t)}$ induces a $(1 + O(\epsilon))$-approximate solution to the dual covering LP (4.3). By Lemma 4.6, there exists a positive vector $\tilde{w}^{(t)} \in \mathbb{R}^V_{>0}$ such that $w^{(t)} \le \tilde{w}^{(t)} \le e^{\epsilon}\tilde{w}^{(t)}$ and such that $\pi^{(t)}$ is decreasing in $\tilde{w}^{(t)}$. Let $\tilde{x}^{(t)} = \tilde{w}^{(t)}/\langle \mathbb{1}, \tilde{w}^{(t)} \rangle$; clearly $\pi^{(t)}$ is decreasing in $\tilde{x}^{(t)}$ as well. We have

$$\hat{f}(\tilde{x}) = \langle \tilde{x}^{(t)}, q(\pi^{(t)}) \rangle \ge e^{-\epsilon} \langle x^{(t)}, q(\pi^{(t)}) \rangle \ge e^{-\epsilon} \hat{f}(x^{(t)}) \ge (1 - O(\epsilon))\lambda^\star.$$

By Lemma 4.3, some set $S_\tau = \{v : \tilde{x}^{(t)} \ge \tau\}$ has density at least $\hat{f}(\tilde{x})$. Since the tth peeling order is increasing in $\tilde{x}^{(t)}$, this set is among the candidate sets $S_{t,j}$ considered by Super-Greedy++. Thus Super-Greedy++ returns a $(1 - O(\epsilon))$-approximate densest subgraph, completing the proof of Theorem 4.1.

5 Densest at-least k Supermodular Subset (DalkSS)

In this section we consider the DalkSS problem, which generalizes DalkSG. Recall that DalkSG is the constrained version of DSG that is much more tractable than other constrained versions, DkSG and DamkSG. The input to the DalkSS problem is a nonnegative monotone supermodular function $f : 2^V \to \mathbb{R}_{\ge 0}$ and a positive integer k. The goal is to find a subset S of the ground set V of size at least k that maximizes the ratio $\frac{f(S)}{|S|}$. Khuller and Saha gave two different $\frac{1}{2}$-approximations for DalkSG. We extend them to DalkSS. These two results are based on having access to exact algorithms for certain subroutines; we also analyze the setting with approximation subroutines. We then consider a natural greedy peeling algorithm for DalkSS in Section 5.3, and prove that it yields a $1/(c_f + 1)$-approximation; this generalizes the algorithm and analysis in [3] for DalkSG. Throughout the section, we use $\rho_f(S) \coloneqq \frac{f(S)}{|S|}$ to simplify notation.

Combinatorial-DalkSS($f : 2^V \to \mathbb{R}_{\geq 0}, k \in \mathbb{Z}_+$)
  $D_0 \leftarrow \emptyset$; $f_0 \leftarrow f$; $j \leftarrow 0$
  while $|D_j| < k$ **do**
    $j \leftarrow j + 1$
    $H_j \leftarrow \arg\max_{S \subseteq V \setminus D_{j-1}} \rho_{f_{j-1}}(S)$
    $D_j \leftarrow D_{j-1} \cup H_j$
    Let $f_j = f_{D_j}$ for notational simplicity
  end while
  Let $\ell \leftarrow j$
  for $j = 1$ to ℓ **do**
    add an arbitrary set of $\max(k - |D_j|, 0)$ vertices to D_j to form D'_j
  end for
  $j^* \leftarrow \arg\max_{j \in [\ell]} \rho_f(D'_j)$
  return D'_{j^*}

Figure 3: A combinatorial approximation algorithm for DalkSS.

5.1 A combinatorial $\frac{1}{2}$-approximation for DalkSS We describe the natural extension of a combinatorial algorithm for DalkSG by Khuller and Saha [37]. Given a supermodular function $f : 2^V \to \mathbb{R}$ and $A \subset V$ we obtain a new function $f_A : 2^{V \setminus A} \to \mathbb{R}$ by *contracting* A; that is, we define $f_A(S) \coloneqq f(A \cup S) - f(A) = f(S \mid A)$ for every $S \subseteq V \setminus A$. It is easy to verify that f_A is also supermodular. Moreover, f_A is normalized and non-negative if f is. The algorithm is based on repeatedly finding a densest subset, contracting, and iterating until the accumulated set achieves the desired size of k. It then returns the best set among the iterations after suitably augmenting it to reach the desired size k. A formal description of this simple and intuitive algorithm is given in Figure 3. The algorithm assumes that an exact algorithm is available for DSS. We discuss extensions using approximation algorithms after the cleaner analysis with the exact algorithm. The algorithm naturally creates a *nested* decomposition of V that has useful properties, which we discuss after the analysis.

Theorem 5.1. *Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a nonnegative monotone supermodular function with $f(\emptyset) = 0$ and $k \in \mathbb{Z}_+$. Then* Combinatorial-DalkSS(f, k) *is a $\frac{1}{2}$-approximation for* DalkSS.

Proof. Let H^* be a densest subset of size at least k. Let ℓ be the total number of iterations in which the while loop of Combinatorial-DalkSS(f, k) is entered. Note that $|D_\ell| \geq k$ and $|D_j| < k$ for all $j < \ell$. We consider two cases.

Case 1: There exists $j \in [\ell - 1]$ such that $f(D_j \cap H^*) \geq \frac{f(H^*)}{2}$. In other words, half the value of H^* is already captured by D_j and $|D_j| < k$; we can add $k - |D_j|$ arbitrary elements to D_j, and due to monotonicity, this will not decrease its value. Formally, we have

$$\rho_f(D'_j) = \frac{f(D'_j)}{k} \geq \frac{f(D_j \cap H^*)}{k} \geq \frac{f(H^*)}{2k} \geq \frac{f(H^*)}{2\,|H^*|} = \frac{\rho_f(H^*)}{2},$$

where the first inequality is by the monotonicity of f, the second inequality is by assumption, and the third inequality uses the fact that $k \leq |H^*|$. Since D'_j is one of the sets considered in the return statement, the algorithm yields a $\frac{1}{2}$-approximation in this case.

Case 2: For every $j \in [\ell - 1]$, $f(D_j \cap H^*) < \frac{f(H^*)}{2}$. Note that we also have $f(D_0 \cap H^*) < \frac{f(H^*)}{2}$ as $D_0 = \emptyset$ and $f(\emptyset) = 0$. For $j \in \{0, 1, \ldots, \ell - 1\}$, let $S_j = V \setminus D_j$. Since $f(D_j \cap H^*) < \frac{f(H^*)}{2}$, informally speaking, there is at least $\frac{f(H^*)}{2}$ value in $V \setminus D_j$, and hence finding a dense set in it with respect to f_j implies that H_{j+1} has density at least $\rho_f(H^*)/2$. Repeating this allows us to argue that the final set D_ℓ has good density. We formalize this now.

We have $D_j \cup H^* = D_j \cup (S_j \cap H^*)$. Via the supermodularity of f, we have $f(D_j \cap H^*) + f(D_j \cup (S_j \cap H^*)) - f(D_j) \geq f(H^*)$. Combining this with the assumption that $f(D_j \cap H^*) < \frac{f(H^*)}{2}$, we have

$$f(D_j \cup (S_j \cap H^*)) - f(D_j) \geq \frac{f(H^*)}{2}.$$

Let $H'_{j+1} = S_j \cap H^*$. Then

$$\rho_{f_j}(H'_{j+1}) = \frac{f_j(H'_{j+1})}{|H'_{j+1}|} = \frac{f(D_j \cup (S \cap H^*)) - f(D_j)}{|H'_{j+1}|} \geq \frac{f(H^*)}{2\,|H'_{j+1}|} \geq \frac{f(H^*)}{2\,|H^*|} = \frac{\rho_f(H^*)}{2}.$$

Since H_{j+1} is a densest set in $V \setminus D_j$ with respect to f_j, we have $\rho_{f_j}(H_{j+1}) \geq \rho_{f_j}(H'_{j+1}) \geq \frac{1}{2} \cdot \rho_f(H^*)$.

We now analyze $\rho_f(D_\ell)$. Recall that $f(\emptyset) = 0$ and hence

$$f(D_\ell) = \sum_{j=0}^{\ell-1} f(D_{j+1}) - f(D_j) = \sum_{j=0}^{\ell-1} f(D_j \cup H_{j+1}) - f(D_j) = \sum_{j=0}^{\ell-1} \rho_{f_j}(H_{j+1}) \cdot |H_{j+1}|\,.$$

As $\rho_{f_j}(H_{j+1}) \geq \frac{1}{2} \cdot \rho_f(H^*)$ for all $j \in \{0, 1, \ldots, \ell-1\}$, we have

$$\sum_{j=0}^{\ell-1} \rho_{f_k}(H_{j+1}) \cdot |H_{j+1}| \geq \frac{\rho_f(H^*)}{2} \cdot \sum_{j=0}^{\ell-1} |H_{j+1}| = \frac{\rho_f(H^*)}{2} \cdot |D_\ell|\,.$$

Thus, $\rho_f(D_\ell) \geq \frac{1}{2} \cdot \rho_f(H^*)$. Since $|D_\ell| \geq k$, it follows $D_\ell = D'_\ell$. As D'_ℓ is considered in the return statement of the algorithm, the algorithm is a $\frac{1}{2}$-approximation in this case. ∎

REMARK 5.1. *The preceding analysis can be easily modified to show that if we used a β-approximation for* DSS *in each iteration where $\beta \in (0, 1]$, then the approximation ratio of* Combinatorial-DalkSS *will be $\frac{\beta}{\beta+1}$. In the first case, instead of considering $f(D_j) \geq \frac{f(H^*)}{2}$ we consider $f(D_j) \geq \frac{\beta}{\beta+1} f(H^*)$. Note that the supermodular function changes in each iteration (obtained via contraction of the original one) and hence the β-approximation needs to apply to each of those.*

Nested decomposition of dense subgraphs. The algorithm Combinatorial-DalkSS$(f, |V|)$ creates a nested decomposition of V into $\emptyset = D_0 \subsetneq D_1 \subsetneq \cdots \subsetneq D_\ell = V$ for some ℓ. It is easy to show, via supermodularity, that there is a unique maximal densest subset; it follows from the fact that if S, T both have density λ^* then $S \cup T$ also has density λ^*. Thus, if one chooses in each iteration j of Combinatorial-DalkSS$(f, |V|)$, a maximal densest subset for f_j from $V \setminus D_{j-1}$, then one obtains a maximal decomposition. Tatti and Gionis [60] and Tatti [59] defined the notion of a *locally dense decomposition* in the context of DSG, and this coincides with the maximal decomposition. The definitions in [59, 60] are not as transparent because the operation of contraction, which is very natural in the context of supermodular functions, can be more involved for specific subclasses since the function after contraction may not be in the same class as the original one. In the context of DSG, Danisch et al. [17] defined the notation of a quotient graph which coincides with the contraction operation, and show that this is equivalent to the definitions in [59, 60]. We refer the reader to [17, 59] for other aspects of (approximate) locally dense decompositions. However, as we saw in Remark 5.1, even with an approximate DSS algorithm that does not necessarily find maximally dense sets, the nested decomposition that is obtained has the following nice property. Given the decomposition, we can compute, for *any* given k, a $\beta/(\beta+1)$-approximation for DALKSS easily. One can view the nested decomposition as an ordering of the vertices which is easy to store. In this sense it has strong similarities to the peeling algorithm which also provides an ordering from which one can obtain an approximation for DALKSS for any given k; see Section 5.3.

5.2 $\frac{1}{2}$-approximation via the natural convex programming relaxation We give a $\frac{1}{2}$-approximation for DALKSS via a natural extension of the LP-based $\frac{1}{2}$-approximation for DALKSG given by Khuller and Saha [37]. Recall that we denote the Lovász extension of f as $\hat{f}$. Let ℓ be an integer. We consider the following convex program which is analogous to (4.2) with additional box constraints to enforce the property that the output set has at least ℓ elements.

$$(\hat{\mathcal{P}}(\ell)) \qquad \text{maximize } \hat{f}(x) \text{ over } x \in [0,1]^V \text{ s.t. } \sum_{v \in V} x_v \leq 1 \text{ and } x_v \leq \frac{1}{\ell} \text{ for all } v \in V.$$

Round-DalkSS($f : 2^V \to \mathbb{R}_{\geq 0}, \ell \in \mathbb{Z}_+, x \in [0,1]^n$)
  let $v_1, v_2, \ldots, v_n$ be ordering of V such that $x(v_1) \leq x(v_2) \leq \cdots \leq x(v_n)$
  for $j \in [n]$, let $S_j := \{v_j, v_{j+1}, \ldots, v_n\}$
  $j^* \leftarrow \arg\max_{j \in [n]} \frac{f(S_j)}{\max\{\ell, |S_j|\}}$
  return S_{j^*} with arbitrary set of $\max\{\ell - |S_{j^*}|, 0\}$ elements added to it

Convex-Program-DalkSS($f : 2^V \to \mathbb{R}_{\geq 0}, k \in \mathbb{Z}_+$)
  for ℓ from k to n **do**
    $x^{(\ell)} \leftarrow$ optimal solution to $(\hat{\mathcal{P}}(\ell))$
    $S^{(\ell)} \leftarrow$ Round-DalkSS$(f, \ell, x^{(\ell)})$
  end for
  $\ell^* \leftarrow \arg\max_{\ell \in \{k, k+1, \ldots, n\}} \rho_f(S^{(\ell)})$
  return $S^{(\ell^*)}$

Figure 4: An approximation for DalkSS via rounding the relaxation $(\hat{\mathcal{P}}(\ell))$.

We show that this program is in fact a valid relaxation of DalkSS if ℓ is chosen to be the cardinality of an optimum solution, and that we can round feasible solutions while only losing a factor of 2. The rounding algorithm is given in Figure 4. Our goal is to prove the following lemma.

LEMMA 5.1. *Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a nonnegative, monotone supermodular function and $k \in \mathbb{Z}_+$. The algorithm* Convex-Program-DalkSS(f, k) *(see Figure 4) is a $\frac{1}{2}$-approximation for* DalkSS.

The first lemma shows that if we know the size of the optimal solution, call it ℓ, then the optimal value of $(\hat{\mathcal{P}}(\ell))$ is at least the optimal value of DalkSS.

LEMMA 5.2. *Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a nonnegative monotone supermodular function. Let H^* be the subset of size at least k with maximum density and let $\ell = |H^*|$. Let $\mathsf{OPT}(\hat{\mathcal{P}}(\ell))$ be the optimal value of $(\hat{\mathcal{P}}(\ell))$. Then $\mathsf{OPT}(\hat{\mathcal{P}}(\ell)) \geq \rho_f(H^*)$.*

Proof. We construct a feasible solution to $(\hat{\mathcal{P}}(\ell))$ with objective value at least $\rho_f(H^*)$. For this solution, set $x_i = \frac{1}{\ell}$ if $i \in H^*$ and 0 otherwise. Clearly, all constraints of $(\hat{\mathcal{P}}(\ell))$ are satisfied. The value of this solution is

$$\hat{f}(x) = \mathbb{E}_{\lambda \in [0,1]}[f(\{v \in V : x_v > \lambda\})] = \frac{1}{\ell} f(H^*) + \left(1 - \frac{1}{\ell}\right) f(\emptyset) \geq \frac{1}{\ell} f(H^*) = \rho_f(H^*),$$

where the inequality follows from the nonnegativity of f. Thus, $\mathsf{OPT}(\hat{\mathcal{P}}(\ell)) \geq \rho_f(H^*)$. ∎

Next, we prove that we can round a solution to $(\hat{\mathcal{P}}(\ell))$ for any ℓ while only losing a factor of 2. The rounding algorithm Round-DalkSS is given in Figure 4.

LEMMA 5.3. *Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a nonnegative monotone supermodular function, let $\ell \geq 1$, and let $x \neq 0$ be a feasible solution to $(\hat{\mathcal{P}}(\ell))$. Let S be the output of the algorithm* Round-DalkSS(f, ℓ, x). *Then $|S| \geq \ell$ and $\rho_f(S) \geq \frac{1}{2}\hat{f}(x)$.*

Proof. For $r \in [0, \frac{1}{\ell}]$, define $S(r) := \{v \in V : x_v \geq r\}$. We show there exists r such that $\frac{f(S(r))}{\max\{\ell, |S(r)|\}} \geq \frac{1}{2}\hat{f}(x)$. Let $r_{\min}$ be the smallest value in $[0, \frac{1}{\ell}]$ such that $|S(r_{\min})| = \ell$. Note then that $|S(r)| \geq \ell$ for $r \leq r_{\min}$.

We claim there exists $r_1 \in [0, r_{\min})$ such that $\rho_f(S(r_1)) \geq \frac{1}{2}\hat{f}(x)$ or there exists $r_2 \in [r_{\min}, \frac{1}{\ell}]$ such that $f(S(r_2)) \geq \frac{\ell}{2} \cdot \hat{f}(x)$. Suppose not. First, we have for $r' \in [0, r_{\min})$, $\rho_f(S(r')) < \frac{1}{2}\hat{f}(x)$. We see that $\int_0^{1/\ell} |S(z)|\, dz = \sum_{v \in V} x_v \leq 1$ as x is feasible for $(\hat{\mathcal{P}}(\ell))$. We have

$$\int_0^{r_{\min}} f(S(z))\, dz < \frac{\hat{f}(x)}{2} \int_0^{r_{\min}} |S(z)|\, dz \leq \frac{\hat{f}(x)}{2} \int_0^{1/\ell} |S(z)|\, dz \leq \frac{\hat{f}(x)}{2}.$$

Second, we have for all $r' \in [r_{\min}, \frac{1}{\ell}]$, $f(S(r')) < \frac{\ell}{2} \cdot \hat{f}(x)$. Then

$$\int_{r_{\min}}^{1/\ell} f(S(z))\,dz < \frac{\ell \cdot \hat{f}(x)}{2} \int_{r_{\min}}^{1/\ell} dz = \frac{\ell \cdot \hat{f}(x)}{2} \left(\frac{1}{\ell} - r_{\min}\right).$$

Combining these two inequalities,

$$\int_{0}^{1/\ell} f(S(z))\,dz < \frac{\hat{f}(x)}{2} + \frac{\ell \cdot \hat{f}(x)}{2} \left(\frac{1}{\ell} - r_{\min}\right) = \hat{f}(x) - \frac{r_{\min} \cdot \ell \hat{f}(x)}{2} \leq \hat{f}(x).$$

This implies that $\int_0^{1/\ell} f(S(z))\,dz < \hat{f}(x)$. As $S(\gamma) = \emptyset$ for all $\gamma > \frac{1}{\ell}$ as x is feasible for $(\hat{\mathcal{P}}(\ell))$, we have $\hat{f}(x) = \int_0^1 f(S(z))\,dz = \int_0^{1/\ell} f(S(z))\,dz$, which is a contradiction.

By the claim, there exists $r \in [0, \frac{1}{\ell}]$ such that $\frac{f(S(r))}{\max\{\ell, |S(r)|\}} \geq \frac{1}{2}\hat{f}(x)$. If we add an arbitrary set of $\max\{\ell - |S(r)|, 0\}$ vertices to $S(r)$ and call this set $S'(r)$, then by the monotonicity of f, we have $\rho_f(S'(r)) \geq \frac{1}{2}\hat{f}(x)$. Assume $0 \leq x(v_1) \leq x(v_2) \leq \cdots \leq x(v_n)$. If there exists i such that $x(v_i) < x(v_{i+1})$, then for all $r_1, r_2 \in [x(v_i), x(v_{i+1}))$, we have $S(r_1) = S(r_2)$. This implies there are at most n distinct sets amongst all of the sets $S(r)$ for $r \in [0, \frac{1}{\ell}]$; these (not necessarily distinct) sets are $S(x(v_1)), S(x(v_2)), \ldots, S(x(v_n))$. As these are all of the sets considered in the algorithm Round-DalkSS, we have that the output of the algorithm S satisfies $\rho_f(S) \geq \frac{1}{2}\hat{f}(x)$. ∎

We are now ready to prove Lemma 5.1.

Proof. [Proof of Lemma 5.1.] Let H^* be a maximum density subset of size at least k. Suppose that we know the size ℓ of H^*. Let $x^{(\ell)}$ be an optimal solution to the concave program $(\hat{\mathcal{P}}(\ell))$. By Lemma 5.3, the output $S^{(\ell)}$ of Round-DalkSS$(f, \ell, x^{(\ell)})$ has size $\left|S^{(\ell)}\right| \geq \ell$ and $\rho_f(S^{(\ell)}) \geq \frac{1}{2}\hat{f}(x^{(\ell)})$. By Lemma 5.2, $\rho_f(S^{(\ell)}) \geq \frac{1}{2}\hat{f}(x^{(\ell)}) \geq \frac{1}{2}\rho_f(H^*)$. As Convex-Program-DalkSS(f, k) checks all possible values of $\ell \geq k$, it follows that the algorithm is a $\frac{1}{2}$-approximation. ∎

We note that a β-approximate solution to $(\hat{\mathcal{P}}(\ell))$, by the above analysis, yields a $\beta/2$-approximate solution to DalkSS.

5.3 Greedy peeling for DalkSS We analyze the natural greedy peeling algorithm for DalkSS. Let $v_1, v_2, \ldots, v_n$ be the order of the elements produced by the natural greedy peeling algorithm for DSS (see Section 3). Let $S_j = \{v_j, v_{j+1}, \ldots, v_n\}$ for $j \in [n]$. The natural greedy peeling algorithm for DalkSS that we analyze in this section returns the set S_j maximizing $\frac{f(S_j)}{|S_j|}$ subject to the constraint that $|S_j| \geq k$. The analysis is inspired by that in [3] for DalkSG; however, we also need Theorem 3.2 on core decomposition for supermodular functions. Recall that $C(\gamma, f)$ is the γ-core of f which we shorten to $C(\gamma)$ when f is clear.

The key technical lemma is the following.

Lemma 5.4. *Let $H \subseteq V$ and let $d = f(H)/|H|$. For any $\alpha \in (0,1)$, $f(C(\alpha d)) \geq (1-\alpha)f(H)$.*

Proof. Let $S_t = C(\alpha d)$. Consider the elements of $H \setminus S_t$ that precede v_t in the peeling order and let them be $v_{j_1}, v_{j_2}, \ldots, v_{j_\ell}$ where $\ell = |H \setminus S_j|$ and $j_1 < j_2 < \cdots < j_\ell < t$. For $i \leq \ell$ we let $A_i = \{v_{j_i}, \ldots, v_{j_\ell}\} \cup (H \cap S_j)$. Note that $A_1 = H$; for notational convenience we let $A_{\ell+1} = H \cap S_j$. By telescoping we see that

$$f(H) = f(A_1) = f(A_{\ell+1}) + \sum_{i=1}^{\ell} (f(A_i) - f(A_{i+1})) = f(A_{\ell+1}) + \sum_{i=1}^{\ell} f(v_{j_i} \mid A_{i+1}).$$

By supermodularity, for each $i \leq \ell$, $f(v_{j_i} \mid A_{i+1}) \leq f(v_{j_i} \mid S_{j_i} - v_{j_i})$ since $A_{i+1} \subseteq S_{j_i} - v_{j_i}$. Since t is the smallest index such that $f(v_t \mid S_t - v_t) \geq \alpha d$, we have $f(v_{j_i} \mid A_{i+1}) < \alpha d$ for $1 \leq i \leq \ell$. Putting together we have

$$f(H) < f(A_{\ell+1}) + \ell \alpha d \leq f(A_{\ell+1}) + \alpha |H| d = f(A_{\ell+1}) + \alpha f(H).$$

Therefore, $f(A_{\ell+1}) \geq (1-\alpha)f(H)$. By monotonicity, $f(S_t) \geq f(A_{\ell+1})$ since $A_{\ell+1} = H \cap S_t$. ∎

THEOREM 5.2. *Greedy peeling yields a $\frac{1}{c_f+1}$-approximation for* DALKSS.

Proof. Let S^* be an optimum solution to the given instance of DALKSS. We have $|S^*| \geq k$. Let $\lambda^* = f(S^*)/|S^*|$ and let $\gamma = \frac{c_f}{c_f+1}\lambda^*$. Let $C(\gamma)$ be the γ-core of f. From Theorem 3.2 we have $C(\gamma) = S_j$ where v_j is the smallest index in the peeling order such that $f(v_j \mid S_j - v_j) \geq \gamma$. We consider two cases.

In the first case, $|S_j| \geq k$, and it is a candidate set considered for output. From the peeling property, $f(u \mid S_j - u) \geq \frac{c_f}{c_f+1}\lambda^*$ for all $u \in S_j$ and hence $f(S_j) \geq \frac{1}{c_f}\sum_{u \in S_j} f(u \mid S_j - u) \geq \frac{1}{c_f}(|S_j| \cdot \frac{c_f}{c_f+1}\lambda^*) = \frac{|S_j|\lambda^*}{c_f+1}$. Therefore $f(S_j)/|S_j| \geq \lambda^*/(c_f+1)$.

In the second case, $|S_j| < k$. Applying Lemma 5.4 to S^* and setting $\alpha = c_f/(c_f+1)$, we have $f(S_j) \geq (1-\alpha)f(S^*) = f(S^*)/(c_f+1)$. Consider S_{n-k+1} which contains S_j since $|S_j| = n - j + 1 < k$. We have $|S_{n-k+1}| = k$ and it is a candidate set considered for output. By monotonicity $f(S_{n-k+1}) \geq f(S_j)$ and hence $\frac{f(S_{n-k+1})}{|S_{n-k+1}|} \geq \frac{f(S_j)}{k} \geq \frac{1}{c_f+1}\frac{f(S^*)}{k} \geq \frac{1}{c_f+1}\frac{f(S^*)}{|S^*|}$ where the last inequality is from the fact that $|S^*| \geq k$. ∎

REMARK 5.2. *Peeling yields a $\frac{1}{c_f}$ approximation for* DSS. *One might ask if the analysis in Section 5.1 (see Remark 5.1) immediately implies that peeling yields a $\frac{1}{(c_f+1)}$-approximation for* DALKSS. *Although this seems plausible, there is a technicality since we cannot show that $c_{f_A} \leq c_f$ for any $A \subseteq V$. This is true for some functions but does not appear to be true for all functions. For this reason we use a direct argument.*

Greedy peeling algorithm for maximizing $\frac{f(S)}{g(|S|)}$ for concave g. Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a nonnegative monotone supermodular function. Let $g : \mathbb{R}_{\geq 0} \to \mathbb{R}_+$ be a non-negative concave function such that $g(0) = 0$. We want to maximize the density $\frac{f(S)}{g(|S|)}$ over all non-empty S. Consider the following algorithm. Let $v_1, v_2, \ldots, v_n$ be the ordering produced by the natural greedy peeling algorithm for the DSS on input f (see Section 3). Let $S_j = \{v_j, v_{j+1}, \ldots, v_n\}$ for $j \in [n]$. The algorithm we consider in this section returns the set S_j maximizing $\frac{f(S_j)}{g(|S_j|)}$. The analysis below generalizes the one in [35] to the supermodular setting via Theorem 5.2.

LEMMA 5.5. *Let $f : 2^V \to \mathbb{R}_{\geq 0}$ be a nonnegative monotone supermodular function where $f(\emptyset) = 0$ and let g be a concave function such that $g(0) = 0$. The greedy peeling algorithm described above for the density $\frac{f(S)}{g(|S|)}$ has an approximation ratio of at least $\frac{1}{c_f+1}$.*

Proof. Let $S^* = \arg\max_{S \subseteq V} \frac{f(S)}{g(|S|)}$ and let $k^* = |S^*|$. Let $v_1, v_2, \ldots, v_n$ be the ordering of the elements produced by greedy peeling for DSS on input f and let $\bar{S}_j = \{v_{n-j+1}, v_{n-j+2}, \ldots, v_n\}$. Note $|\bar{S}_j| = j$.

Let $i = \arg\max_{j:|\bar{S}_j| \geq k^*} f(\bar{S}_j)/|\bar{S}_j|$. We will prove that the set $\bar{S}_i$ yields a good approximation.

We have

$$\frac{f(S^*)}{g(|S^*|)} = \frac{f(S^*)}{|S^*|} \cdot \frac{|S^*|}{g(|S^*|)} \leq (c_f+1)\frac{f(\bar{S}_i)}{|\bar{S}_i|} \cdot \frac{|S^*|}{g(|S^*|)} \leq (c_f+1)\frac{f(\bar{S}_i)}{|\bar{S}_i|} \cdot \frac{|\bar{S}_i|}{g(|\bar{S}_i|)} = (c_f+1)\frac{f(\bar{S}_i)}{g(|\bar{S}_i|)}$$

where the first inequality is by Theorem 5.2 and definition of i, and the second inequality is due to concavity of g and the fact that $|\bar{S}_i| \geq |S^*|$. This concludes the proof. ∎

References

[1] Zeyuan Allen-Zhu and Lorenzo Orecchia. "Nearly linear-time packing and covering LP solvers". In: *Mathematical Programming* 175.1 (2019), pp. 307–353 (cit. on p. 3).

[2] Noga Alon, Sanjeev Arora, Rajsekar Manokaran, Dana Moshkovitz, and Omri Weinstein. "Inapproximability of densest κ-subgraph from average case hardness". In: *Unpublished manuscript* 1 (2011) (cit. on p. 6).

[3] Reid Andersen and Kumar Chellapilla. "Finding dense subgraphs with size bounds". In: *International Workshop on Algorithms and Models for the Web-Graph*. Springer. 2009, pp. 25–37 (cit. on pp. 1, 4, 5, 16, 20).

[4] Sanjeev Arora, Elad Hazan, and Satyen Kale. "The multiplicative weights update method: a meta-algorithm and applications". In: *Theory of Computing* 8.1 (2012), pp. 121–164 (cit. on p. 14).

[5] Yuichi Asahiro, Kazuo Iwama, Hisao Tamaki, and Takeshi Tokuyama. "Greedily finding a dense subgraph". In: *Journal of Algorithms* 34.2 (2000), pp. 203–221 (cit. on p. 2).

[6] Kyriakos Axiotis, Adam Karczmarz, Anish Mukherjee, Piotr Sankowski, and Adrian Vladu. "Decomposable Submodular Function Minimization via Maximum Flow". In: *arXiv preprint arXiv:2103.03868* (2021) (cit. on p. 9).

[7] Bahman Bahmani, Ashish Goel, and Kamesh Munagala. "Efficient primal-dual graph algorithms for mapreduce". In: *International Workshop on Algorithms and Models for the Web-Graph*. Springer. 2014, pp. 59–78 (cit. on pp. 6, 12).

[8] Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. "Densest subgraph in streaming and MapReduce". In: *Proceedings of the VLDB Endowment* 5.5 (2012), pp. 454–465 (cit. on p. 6).

[9] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. "Detecting high log-densities: an $O(n^{1/4})$ approximation for densest k-subgraph". In: *Proceedings of the forty-second ACM symposium on Theory of computing*. 2010, pp. 201–210 (cit. on pp. 1, 6).

[10] Aditya Bhaskara, Moses Charikar, Venkatesan Guruswami, Aravindan Vijayaraghavan, and Yuan Zhou. "Polynomial integrality gaps for strong SDP relaxations of densest k-subgraph". In: *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*. SIAM. 2012, pp. 388–405 (cit. on p. 6).

[11] Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos Tsourakakis. "Space-and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams". In: *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. 2015, pp. 173–182 (cit. on p. 6).

[12] Digvijay Boob, Yu Gao, Richard Peng, Saurabh Sawlani, Charalampos Tsourakakis, Di Wang, and Junxing Wang. "Flowless: Extracting Densest Subgraphs Without Flow Computations". In: *Proceedings of The Web Conference 2020*. 2020, pp. 573–583 (cit. on pp. 1, 3, 4, 6, 7, 11, 12).

[13] Digvijay Boob, Saurabh Sawlani, and Di Wang. "Faster width-dependent algorithm for mixed packing and covering LPs". In: *Advances in Neural Information Processing Systems 32 (NIPS 2019)* (2019) (cit. on pp. 1, 3, 6, 12).

[14] Moses Charikar. "Greedy approximation algorithms for finding dense components in a graph". In: *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer. 2000, pp. 84–95 (cit. on pp. 2, 3, 6, 7, 12, 13).

[15] Chandra Chekuri and Kent Quanrud. "A fast approximation for maximum weight matroid intersection". In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2016, pp. 445–457 (cit. on p. 5).

[16] Michele Conforti and Gérard Cornuéjols. "Submodular set functions, matroids and the greedy algorithm: tight worst-case bounds and some generalizations of the Rado-Edmonds theorem". In: *Discrete applied mathematics* 7.3 (1984), pp. 251–274 (cit. on p. 4).

[17] Maximilien Danisch, T.-H. Hubert Chan, and Mauro Sozio. "Large scale density-friendly graph decomposition via convex programming". In: *Proceedings of the 26th International Conference on World Wide Web*. 2017, pp. 233–242 (cit. on pp. 5, 6, 18).

[18] Ran Duan and Seth Pettie. "Linear-time approximation for maximum weight matching". In: *Journal of the ACM (JACM)* 61.1 (2014), pp. 1–23 (cit. on p. 5).

[19] Jack Edmonds. "Submodular functions, matroids, and certain polyhedra". In: *Combinatorial Structures and Their Applications (Proceedings Calgary International Conference on Combinatorial Structures and Their Applications, Calgary, Alberta, 1969; R. Guy, H. Hanani, N. Sauer, J. Schönheim, eds.)* Ed. by R. Guy, H. Hanani, N. Sauer, and J. Schönheim. New York: Gordon and Breach, 1970 (cit. on pp. 12, 13).

[20] Alina Ene, Huy L Nguyen, and László A Végh. "Decomposable submodular function minimization: discrete and continuous". In: *arXiv preprint arXiv:1703.01830* (2017) (cit. on p. 9).

[21] Hossein Esfandiari, MohammadTaghi Hajiaghayi, and David P Woodruff. "Brief announcement: Applications of uniform sampling: Densest subgraph and beyond". In: *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures*. 2016, pp. 397–399 (cit. on p. 6).

[22] András Faragó. "A general tractable density concept for graphs". In: *Mathematics in Computer Science* 1.4 (2008), pp. 689–699 (cit. on pp. 2, 9).

[23] András Faragó and Zohre R Mojaveri. "In search of the densest subgraph". In: *Algorithms* 12.8 (2019), p. 157 (cit. on p. 1).

[24] Uriel Feige. "Relations between average case complexity and approximation complexity". In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 2002, pp. 534–543 (cit. on p. 6).

[25] Uriel Feige, David Peleg, and Guy Kortsarz. "The dense k-subgraph problem". In: *Algorithmica* 29.3 (2001), pp. 410–421 (cit. on p. 1).

[26] Giorgio Gallo, Michael D Grigoriadis, and Robert E Tarjan. "A fast parametric maximum flow algorithm and applications". In: *SIAM Journal on Computing* 18.1 (1989), pp. 30–55 (cit. on p. 2).

[27] Mohsen Ghaffari, Silvio Lattanzi, and Slobodan Mitrović. "Improved parallel algorithms for density-based network clustering". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2201–2210 (cit. on pp. 1, 6).

[28] Aristides Gionis and Charalampos E Tsourakakis. "Dense subgraph discovery: Kdd 2015 tutorial". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, pp. 2313–2314 (cit. on p. 1).

[29] Andrew V Goldberg. *Finding a maximum density subgraph*. University of California Berkeley, 1984 (cit. on pp. 2, 7, 12).

[30] Andrew V. Goldberg and Robert E. Tarjan. "Finding Minimum-Cost Circulations by Successive Approximation". In: *Math. Oper. Res.* 15.3 (1990), pp. 430–466. Preliminary verison in STOC, 1987 (cit. on p. 9).

[31] Shuguang Hu, Xiaowei Wu, and TH Hubert Chan. "Maintaining densest subsets efficiently in evolving hypergraphs". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 929–938 (cit. on p. 2).

[32] Victor P. Il'ev. "An approximation guarantee of the greedy descent algorithm for minimizing a supermodular set function". In: *Discrete Applied Mathematics* 114.1 (2001). discrete analysis & operations research, pp. 131–146 (cit. on p. 4).

[33] Martin Jaggi. "Revisiting Frank-Wolfe: Projection-free sparse convex optimization". In: *International Conference on Machine Learning*. PMLR. 2013, pp. 427–435 (cit. on p. 6).

[34] Ravindran Kannan and V Vinay. "Analyzing the structure of large graphs". Unpublished manuscript. 1999 (cit. on pp. 3, 6).

[35] Yasushi Kawase and Atsushi Miyauchi. "The densest subgraph problem with a convex/concave size function". In: *Algorithmica* 80.12 (2018), pp. 3461–3480 (cit. on pp. 4, 5, 10, 21).

[36] Subhash Khot. "Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique". In: *SIAM Journal on Computing* 36.4 (2006), pp. 1025–1071 (cit. on pp. 1, 6).

[37] Samir Khuller and Barna Saha. "On finding dense subgraphs". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2009, pp. 597–608 (cit. on pp. 1, 3–6, 9, 17, 18).

[38] Guy Kortsarz and David Peleg. "Generating sparse 2-spanners". In: *Journal of Algorithms* 17.2 (1994), pp. 222–236 (cit. on pp. 2, 4).

[39] Guy Kortsarz and David Peleg. "On Choosing a Dense Subgraph (Extended Abstract)". In: *34th Annual Symposium on Foundations of Computer Science, Palo Alto, California, USA, 3-5 November 1993*. IEEE Computer Society, 1993, pp. 692–701 (cit. on p. 1).

[40] E.L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, 1976 (cit. on p. 2).

[41] Victor E Lee, Ning Ruan, Ruoming Jin, and Charu Aggarwal. "A survey of algorithms for dense subgraph discovery". In: *Managing and Mining Graph Data*. Springer, 2010, pp. 303–336 (cit. on p. 1).

[42] László Lovász. "Submodular functions and convexity". In: *Mathematical programming the state of the art*. Springer, 1983, pp. 235–257 (cit. on pp. 12, 13).

[43] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks V.S. Lakshmanan, Wenjie Zhang, and Xuemin Lin. "Efficient Algorithms for Densest Subgraph Discovery on Large Directed Graphs". In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. SIGMOD '20. Portland, OR, USA: Association for Computing Machinery, 2020, 1051–1066 (cit. on pp. 1, 3, 6).

[44] Fragkiskos D Malliaros, Christos Giatsidis, Apostolos N Papadopoulos, and Michalis Vazirgiannis. "The core decomposition of networks: Theory, algorithms and applications". In: *The VLDB Journal* 29.1 (2020), pp. 61–92 (cit. on pp. 3, 4, 11).

[45] Pasin Manurangsi. "Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. 2017, pp. 954–961 (cit. on pp. 1, 6).

[46] Pasin Manurangsi. "Inapproximability of maximum biclique problems, minimum k-cut and densest at-least-k-subgraph from the small set expansion hypothesis". In: *Algorithms* 11.1 (2018), p. 10 (cit. on pp. 1, 5).

[47] Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T. Vu. "Densest Subgraph in Dynamic Graph Streams". In: *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II*. Ed. by Giuseppe F. Italiano, Giovanni Pighizzini, and Donald Sannella. Vol. 9235. Lecture Notes in Computer Science. Springer, 2015, pp. 472–482 (cit. on p. 6).

[48] Michael Mitzenmacher, Jakub Pachocki, Richard Peng, Charalampos Tsourakakis, and Shen Chen Xu. "Scalable large near-clique detection in large-scale networks via sampling". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, pp. 815–824 (cit. on p. 6).

[49] Robert Nishihara, Stefanie Jegelka, and Michael I Jordan. "On the convergence rate of decomposable submodular function minimization". In: *arXiv preprint arXiv:1406.6474* (2014) (cit. on p. 9).

[50] Lorenzo Orecchia and Zeyuan Allen Zhu. "Flow-Based Algorithms for Local Graph Clustering". In: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*. Ed. by Chandra Chekuri. SIAM, 2014, pp. 1267–1286 (cit. on p. 5).

[51] Jean-Claude Picard and Maurice Queyranne. "A network flow solution to some nonlinear 0-1 programming problems, with applications to graph theory". In: *Networks* 12.2 (1982), pp. 141–159 (cit. on pp. 2, 7, 12).

[52] Serge A Plotkin, David B Shmoys, and Éva Tardos. "Fast approximation algorithms for fractional packing and covering problems". In: *Mathematics of Operations Research* 20.2 (1995), pp. 257–301 (cit. on p. 14).

[53] Kent Quanrud. "Fast approximations for combinatorial optimization via multiplicative weight updates". PhD thesis. University of Illinois, Urbana-Champaign, 2019 (cit. on p. 3).

[54] Saurabh Sawlani and Junxing Wang. "Near-optimal fully dynamic densest subgraph". In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. 2020, pp. 181–193 (cit. on pp. 1, 3, 6, 9).

[55] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Vol. 24. Springer Science & Business Media, 2003 (cit. on pp. 2, 13).

[56] Peter Stobbe and Andreas Krause. "Efficient minimization of decomposable submodular functions". In: *arXiv preprint arXiv:1010.5511* (2010) (cit. on p. 9).

[57] Bintao Sun, Maximilien Danisch, TH Chan, and Mauro Sozio. "KClist++: A Simple Algorithm for Finding k-Clique Densest Subgraphs in Large Graphs". In: *Proceedings of the VLDB Endowment (PVLDB)* (2020) (cit. on pp. 5, 6).

[58] Maxim Sviridenko, Jan Vondrák, and Justin Ward. "Optimal approximation for submodular and supermodular optimization with bounded curvature". In: *Mathematics of Operations Research* 42.4 (2017), pp. 1197–1218 (cit. on p. 4).

[59] Nikolaj Tatti. "Density-friendly graph decomposition". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 13.5 (2019), pp. 1–29 (cit. on pp. 5, 18).

[60] Nikolaj Tatti and Aristides Gionis. "Density-friendly graph decomposition". In: *Proceedings of the 24th International Conference on World Wide Web*. 2015, pp. 1089–1099 (cit. on pp. 5, 18).

[61] Charalampos Tsourakakis. "The k-clique densest subgraph problem". In: *Proceedings of the 24th international conference on world wide web*. 2015, pp. 1122–1132 (cit. on pp. 2, 9).

[62] Charalampos Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria Tsiarli. "Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees". In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, pp. 104–112 (cit. on p. 5).

[63] Charalampos Tsourakakis and Tianyi Chen. *Dense subgraph discovery: Theory and application (Tutoral at SDM 2021)*. `https://tsourakakis.com/dense-subgraph-discovery-theory-and-applications-tutorial-sdm-2021/`. 2021 (cit. on p. 1).

[64] Charalampos E Tsourakakis. "A novel approach to finding near-cliques: The triangle-densest subgraph problem". In: *arXiv preprint arXiv:1405.1477* (2014) (cit. on p. 2).

[65] Nate Veldt, Austin R. Benson, and Jon Kleinberg. *The Generalized Mean Densest Subgraph Problem*. 2021. arXiv: `2106.00909 [cs.DS]` (cit. on pp. 1, 2, 5, 10).

[66] Di Wang. "Fast Approximation Algorithms for Positive Linear Programs". PhD thesis. EECS Department, University of California, Berkeley, 2017 (cit. on p. 3).

[67] Di Wang, Satish Rao, and Michael W Mahoney. "Unified Acceleration Method for Packing and Covering Problems via Diameter Reduction". In: *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 2016 (cit. on p. 3).