Supermodular Maximization with Cardinality Constraints*

Xujin Chen Xiaodong Hu Changjun Wang

Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China

School of Mathematical Sciences. University of Chinese Academy of Sciences, Beijing 100049, China

{xchen,xdhu, wcj}@amss.ac.cn

Qingjie Ye

School of Mathematical Sciences, Key Laboratory of MEA (Ministry of Education),

Shanghai Key Laboratory of PMMP, East China Normal University, Shanghai, 200241, China

qjye@math.ecnu.edu.cn

Abstract

Let V be a finite set of n elements, $f : 2^V \to \mathbb{R}_+$ be a nonnegative monotone supermodular function, and k be a positive integer no greater than n. This paper addresses the problem of maximizing $f(S)$ over all subsets $S \subseteq V$ subject to the cardinality constraint $|S| = k$ or $|S| \leq k$.

Let r be a constant integer. The function f is assumed to be *r-decomposable*, meaning there exist $m\,(\geq 1)$ subsets $V_1, \ldots, V_m$ of V, each with a cardinality at most r, and a corresponding set of nonnegative supermodular functions $f_i : 2^{V_i} \to \mathbb{R}_+$, $i = 1, \ldots, m$ such that $f(S) = \sum_{i=1}^{m} f_i(S \cap V_i)$ holds for each $S \subseteq V$. Given r as an input, we present a polynomial-time $O(n^{(r-1)/2})$-approximation algorithm for this maximization problem, which does not require prior knowledge of the specific decomposition.

When the decomposition $(V_i, f_i)_{i=1}^m$ is known, an additional connectivity requirement is introduced to the problem. Let G be the graph with vertex set V and edge set $\cup_{i=1}^m \{uv : u, v \in V_i, u \neq v\}$. The cardinality constrained solution set S is required to induce a connected subgraph in G. This model generalizes the well-known problem of finding the densest connected k-subgraph. We propose a polynomial time $O(n^{(r-1)/2})$-approximation algorithm for this generalization. Notably, this algorithm gives an $O(n^{1/2})$-approximation for the densest connected k-subgraph problem, improving upon the previous best-known approximation ratio of $O(n^{2/3})$.

Keywords: Supermodular function, Cardinality constraint, Approximation algorithm, Connectivity

1 Introduction

Given a function $f : 2^V \to \mathbb{R}$ over subsets of a finite ground set V, for any $v \in V$ and $S \subseteq V$, let $f(v|S) = f(S \cup \{v\}) - f(S)$ denote the *marginal value* of v with respect to S. Function f is *supermodular* if $f(A) + f(B) \leq f(A \cup B) + f(A \cap B)$ for all $A, B \subseteq V$, or equivalently $f(v|B) \leq f(v|A)$ for all $B \subseteq A$ and $v \in V \setminus A$. This paper adopts the value-oracle model: given any $S \subseteq V$, a black-box oracle returns the value $f(S)$.

Supermodular functions play a fundamental role in both theoretical studies and practical applications. Their theoretical significance is evident in several well-established results, such as the fact that a set function is supermodular if and only if its Lovász extension is concave [38], and that any set function can be represented as the difference between two supermodular functions [40]. Moreover, supermodular functions have found extensive applications across diverse fields. For example, they are used in the analysis of economic systems and game theory to model increasing returns and strategic complementarity [52], and in the design of algorithms

*Research supported in part by the National Natural Science Foundation of China (NSFC) under grant numbers 11971046, 12331014 and 71871009; Science and Technology Commission of Shanghai Municipality (No. 22DZ2229014)

for optimization problems such as scheduling [13], resource allocation [44], and supply chain management [29].

Cardinality constraints. Many real-world applications involve maximization over a nonnegative supermodular function $f : 2^V \to \mathbb{R}_+$ subject to a cardinality constraint. Examples include public-key encryption, computational biology, data sampling, and document summarization (see, e.g., [2, 4, 7]). Suppose that the ground set V comprises n elements, and k is a positive integer not exceeding n. A subset S of V is classified as: a *k-subset* if $|S| = k$, a *$\underline{k}$-subset* if $|S| \leq k$, and a *$\overline{k}$-subset* if $|S| \geq k$. The constrained maximization typically involves finding either a k-subset or a $\underline{k}$-subset S of V that maximizes $f(S)$. The problem is referred to as the *k-subset supermodular maximization* (*kSPM*) problem in the former case, and as the *$\underline{k}$SPM* problem in the latter case.

If f is monotone, i.e., $f(B) \leq f(A)$ for all $B \subseteq A$, then the $\underline{k}$SPM problem on f is equivalent to the kSPM problem, in that an optimal solution of one problem can be converted to an optimal solution of the other with the same objective value.

Decomposability. In practice, the supermodular function f often exhibits certain structural properties related to decomposition, which helps to obtain better approximations for supermodular maximization [14]. Let r be a positive integer. A supermodular function $f : 2^V \to \mathbb{R}_+$ is *r-decomposable* if there exist subsets $V_1, \ldots, V_m$ of V and supermodular functions $f_i : 2^{V_i} \to \mathbb{R}_+$, $i = 1, \ldots, m$ such that $|V_i| \leq r$ for each i and $f(S) = \sum_{i=1}^m f_i(S \cap V_i)$ for each $S \subseteq V$. We call the corresponding decomposition a *r-decomposition*, and denote it as $(r; (V_i, f_i)_{i=1}^m)$, or $(V_i, f_i)_{i=1}^m$ if r is clear from the context.

Trivially, r could be taken as n, for which the decomposition is (V, f). Generally, the smaller r is, the better the structural properties f has, and the more algorithmically amenable the maximization on f becomes. As a key example, it is not hard to prove that f is 1-decomposable if and only if f is modular; that is, $f(A) + f(B) = f(A \cup B) + f(A \cap B)$ for all $A, B \subseteq V$. For modular functions, the kSPM and $\underline{k}$SPM problems simplify to selecting the top k most valuable elements.

Connectivity constraints. Given the decomposition $(V_i, f_i)_{i=1}^m$ for supermodular function f, we construct an undirected simple graph $G = (V, E)$ with vertex set V and edge set $E := \cup_{i=1}^m \{uv : \{u, v\} \subseteq V_i, u \neq v\}$. Every nonempty subset S of V *induces* a subgraph of G, denoted by $G[S]$, whose vertex set is S and whose edge set consists of edges in E with both end-vertices in S. We call the set S *connected* if the induced graph $G[S]$ is connected. Imposing connectivity requirements on subsets arises naturally in many applications (see, e.g., [2, 20, 33]). This leads to supermodular optimization involving connectivity and cardinality constraints: the *connected k-subset supermodular maximization* (CkSMP) problem for finding a connected k-subset S with maximum $f(S)$, and the *connected $\underline{k}$-subset supermodular maximization* (C$\underline{k}$SMP) problem for finding a connected $\underline{k}$-subset S with maximum $f(S)$.

Unlike the counterpart without connectivity requirement, even if f is monotone, the cardinality constraint $|S| \leq k$ in the C$\underline{k}$SPM problem cannot be tightened to $|S| = k$ because possibly no k-subset is connected. For example, if $|V_i| = 1$ for all $1 \leq i \leq m$, then G is edgeless, and no set of cardinality 2 or more is connected. On the other hand, for monotone functions, the CkSPM and C$\underline{k}$SPM problems are "indirectly" equivalent. This means that the algorithm for one problem can be used to solve the other by examining all connected components of G, without any loss of solution quality or time complexity.

For convenience, throughout the paper, we use $\mathscr{F}_r$ to denote the family of monotone nonnegative supermodular functions that are r-decomposable, where *r is a constant*. The main goal of this paper is to develop polynomial-time $O(n^{(r-1)/2})$-approximation algorithms for the kSPM, $\underline{k}$SPM, and C$\underline{k}$SPM problems on $\mathscr{F}_r$. The algorithms for C$\underline{k}$SPM apply to CkSPM with an easy expansion from a $\underline{k}$-set to a k-set.

1.1 Related work

Maximizing a supermodular function is equivalent to minimizing its negation, i.e., a submodular function. The unconstrained submodular minimization is solvable in (strongly) polynomial time (see [25, 48]). Extensive research has been conducted on the optimization problems associated

with both submodular and supermodular functions. The reader is referred, e.g. to [11, 12, 24, 41, 42, 51] for various algorithmic studies in this area.

Svitkina and Fleischer [51] studied submodular minimization with cardinality constraints. They proved that any algorithm that makes a polynomial number of oracle queries cannot achieve an $o\left(\sqrt{\frac{n}{\ln n}}\right)$-approximation for finding a minimum-valued $\underline{k}$-set. On the other hand, they proposed a $(5\sqrt{\frac{n}{\ln n}}, \frac{1}{2})$-bicriteria approximation algorithm for $\overline{k}$-sets, which outputs a $\overline{k/2}$-set whose value is at most $5\sqrt{\frac{n}{\ln n}}$ times the minimum value of a $\overline{k}$-set. Besides, they showed that there is no (ρ, σ)-bicriteria approximation algorithm that makes a polynomial number of oracle queries, for any ρ and σ with $\frac{\rho}{\sigma} = o(\sqrt{\frac{n}{\ln n}})$.

Bai and Bilmes [7] designed a β_f-approximation algorithm for the kSPM problem when f is monotone, where $\beta_f = \max_{v \in V} f(v|V \setminus \{v\})/f(v)$. Moreover, they proved that for any $\epsilon > 0$, the kSPM problem does not admit any $(\beta_f - \epsilon)$-approximation in polynomial time. However, the ratio β_f is unbounded in a general sense as there may be $v \in V$ such that $f(v) = 0$ and $f(v|V \setminus \{v\}) > 0$ (see Appendix A for an example of $f \in \mathscr{F}_2$). In addition, they proved that for any $\epsilon > 0$, there is no $O(2^{(0.5-\epsilon)n})$-approximation in polynomial time even when considering only monotone functions and allowing randomized algorithms.

To the best of our knowledge, apart from the work by Bai and Bilmes [7], no approximation algorithms were proposed for the general kSPM and CkSPM problems, even under the common assumption that the input nonnegative supermodular function is monotone and r-decomposable. In contrast, a substantial literature addresses special cases of these problems, mainly focusing on instances arising from hypergraphs and ordinary graphs.

Special cases of kSPM. Let $H = (V, E)$ be a hypergraph with vertex set V and hyperedge set E. For any subset S of V, let $E(S) = \{e \in E : e \subseteq S\}$ denote the set of hyperedges in the subhypergraph of H induced by S. The *density* of this subhypergraph, a.k.a. the *density* of S, is defined as $\mu(S) = |E(S)|/|S|$. While the problem of finding a densest subhypergraph is solvable in polynomial time [17, 28], its cardinality-constrained variant is NP-hard. This variant is known as the *densest k-subhypergraph* (DkSH) problem, where one needs to find a k-subset S of V that maximizes the density $\mu(S) = |E(S)|/k$, or equivalently, maximizes the number $|E(S)|$ of hyperedges contained in S. It is obvious that $\mu(S) = |E(S)|/k$ is a supermodular function on 2^V, and therefore, the kSPM problem generalizes the DkSH problem. Besides, the supermodular function μ is nonnegative, monotone, and r-decomposable, where $r = \max\{|e| : e \in E\}$ is often called the *rank* of H. Although $\beta_\mu = +\infty$ in general, researchers managed to obtain better approximations for uniform hypergraphs. A hypergraph is r-uniform if every hyperedge in it consists of exactly r vertices. Chlamtáč et al. [17] designed an $O(n^{0.697831+\epsilon})$-approximation algorithm for DkSH on 3-uniform hypergraphs that runs in $n^{O(1/\varepsilon)}$ time. However, this approximation does not extend to the general rank-3 hypergraphs.

When restricted to 2-uniform hypergraphs, i.e., ordinary graphs G, the DkSH problem reduces to the well-known *densest k-subgraph* (DkS) problem, which was first investigated by Kortsarz and Peleg [34] in 1993. Since then, it has been extensively studied in both directions of designing approximation algorithms and proving inapproximation lower bounds [6, 8–10, 22, 23, 34, 39].

On the positive side, as shown by Feige et al. [22], every approximation algorithm designed for the unweighted version of DkS, in which all edges have weight 1, can be extended to handle the (general) weighted case. This extension introduces an additional $O(\log n)$ factor to the approximation ratio. The approximation ratio for weighted DkS has been improved over the years. Kortsarz and Peleg [34] first provided an $\mathcal{O}(n^{0.3885})$-approximation algorithm that runs in $\mathcal{O}(n^5)$ time, where the notation $\mathcal{O}$ ignores polylogarithmic factors in n. Later, Feige et al. [22] improved the approximation ratio to $O(n^{1/3-\delta})$ for some small $\delta > 0$ (which was estimated to be roughly 1/60). This approximation ratio had remained the best for almost ten years until 2010 when Bhaskara et al. [8] proposed an $O(n^{1/4+\varepsilon})$-approximation algorithm that runs in $n^{O(1/\varepsilon)}$ time. This remains the state-of-the-art approximation ratio for the DkS problem when parameterized solely by n. When the desired subgraph size k is also involved, combining several ideas from binary search, "expanding kernels" and conditional probability, Kortsarz and Peleg [34] gave an $O(n/k)$-approximation algorithm that runs in $O(m \log W)$ time, where m is the number of edges in graph G and W is the total weight of $\min\{m, k^2\}$ edges of highest weights. When $k < n/3$, Asahiro et al. [6] proved the approximation ratio of $\Theta(n/k)$ for the heuristic that repeatedly removes a vertex with the minimum weighted degree until there are k vertices

left. Later, Feige et al. [22] showed that a different procedure using greedy attachments finds an $O(n/k)$-approximation in $O(m + n \log n)$ time. Although linear and semidefinite programming relaxation approaches have been adopted in [23, 26, 49] to design algorithms with approximation ratios somewhat better than $O(n/k)$ for certain value ranges of k, the $O(n/k)$ ratio remains the state-of-the-art for DkS in the general case, applying to all values of n and k for both weighted and unweighted graphs. For some special cases characterized by restrictive graph classes, specific values of parameter k, and particular (weighted) densities of optimal solutions, improved approximations have been achieved [5, 15, 19, 34, 35, 37, 43].

On the negative side, no nontrivial lower bound on the approximability of DkS has been known under the standard assumption of P $\neq$ NP. However, $(1 + \varepsilon)$-inapproximability [21, 31], constant-factor-inapproximability and superconstant-factor-inapproximability [1, 46] were established for (unweighted) DkS under various complexity assumptions. Manurangsi [39] showed that, under the exponential time hypothesis (ETH), no polynomial-time algorithm can approximate DkS to within $n^{1/(\log \log n)^c}$ factor of the optimum, where $c > 0$ is a universal constant independent of n. Recently, Jones et al. [30] provided the hardness evidence for DkS by showing lower bounds against the powerful Sum-of-Squares (SoS) algorithm: for any $\epsilon > 0$, there exists a constant $\delta > 0$ such that degree-n^δ SoS exhibits an integrality gap of $O(n^{1/4-\epsilon})$ for DkS. The gap matches the approximation ratio by Bhaskara et al. [8] mentioned above.

Special case of CkSPM. When graphical connectivity constraint is imposed on the DkS, the problem transforms to a special case of CkSPM. Given a graph $G = (V, E)$ with edge weights $w \in \mathbb{R}_+^E$, this yields a 2-decomposition $(\{u, v\}, w_{uv}/k)_{uv \in E}$ of monotone supermodular function $\mu : 2^V \to \mathbb{R}_+$ with $\mu(\{u, v\}) = w_{uv}$ for all $uv \in E$. A subgraph with k vertices in G is termed a *k-subgraph*. Assuming G is connected, the *densest connected k-subgraph* (DCkS) problem seeks a connected k-subgraph in G of maximum weighted density. This amounts to identifying a connected k-subset S of V with maximum $\mu(S)$.

Compared with the large literature on DkS, the work on approximating the densest connected k-subgraphs is relatively limited. A trivial $O(k)$-approximation can be achieved by examining all stars [34]. Chen et al. [16] designed an $O(nm)$-time algorithm that finds a connected k-subgraph whose weighted density is at least $\Omega(k^2/n^2)$ of the maximum weighted density among all k-subgraphs of G, which are not necessarily connected. It follows that DCkS is approximable within a factor of $O(\min\{k, n^2/k^2\}) = O(n^{2/3})$. For the unweighted version of DCkS, an $O(n^{2/5})$-approximation can be guaranteed by combining the $O(n^2/k^2)$-approximation algorithm with various methods that search for dense connected k-subgraphs based on densest subgraphs, high-degree vertices and 2-hop neighborhoods of vertices [16]. Other works on designing polynomial-time algorithms for DCkS only deal with special graphical topologies, such as trees [18, 45], h-trees, cographs, split graphs [18], complete graphs [27, 47], and interval graphs whose clique graphs are paths [36].

Upper bound of kSPM. The kSPM problem over a supermodular function $f : 2^V \to \mathbb{R}_+$ is related to the *densest at-least-k supermodular subset* (D$\overline{k}$SS) problem. The objective of D$\overline{k}$SS to find a $\overline{k}$-set S of V that maximizes $f(S)/|S|$. The optimal (approximate) solutions to D$\overline{k}$SS provide upper bounds on the objective values of the kSPM problem. Indeed, for any optimal solution S^* of the D$\overline{k}$SS and any k-subset S, it holds that $f(S^*) \geq f(S^*) \cdot \frac{|S|}{|S^*|} \geq \frac{f(S)}{|S|} \cdot |S| = f(S)$. Chekuri et al. [14] established the following result.

Theorem 1.1 (Chekuri et al. [14])**.** *There is an $O(n^2)$ time $(r + 1)$-approximation algorithm for D$\overline{k}$SS on nonnegative supermodular functions that are monotone and r-decomposable.*

Moreover, Chekuri et al. [14] designed a polynomial-time 2-approximation algorithm for D$\overline{k}$SS on monotone nonnegative supermodular functions.

As a special case of D$\overline{k}$SS, the densest at-least-k subgraph (D$\overline{k}$S) problem is to find a densest subgraph with k or more vertices. Khuller and Saha [32] proved that the D$\overline{k}$S problem is NP-hard, and proposed a maximum-flow-based 2-approximation algorithm for D$\overline{k}$S, which runs in $O(n^2 m \log^2 n)$ time. They also presented an LP-based polynomial-time algorithm with the same approximation ratio 2. The algorithm by Andersen and Chellapilla [3] attains an approximation ratio of 3 for D$\overline{k}$S with shorter running time $O(m + n \log n)$.

1.2 Our results

In this paper, we propose $O(n^{(r-1)/2})$-approximation algorithms for supermodular maximization on $\mathscr{F}_r$ with a cardinality constraint or with both cardinality and connectivity constraints. The algorithm for kSPM and $\underline{k}$SPM runs in $O(n^{r+1})$ time, and that for CkSPM and C$\underline{k}$SPM runs in $O(mn^{r+1})$ time. These general settings of constrained supermodular maximization have been less explored in the literature. To the best of our knowledge, the only existing result is the $\max\{f(v|V \setminus \{v\})/f(v) : v \in V\}$-approximation for kSPM [7], where the approximation ratio can be arbitrarily large, even for $f \in \mathscr{F}_2$, and when the size of the ground set is fixed at $n \geq 4$ (see Appendix A).

Notably, our result on CkSPM implies an $O(n^{0.5})$-approximation algorithm for the densest connected k-subgraph (DCkS) problem, improving the ratio in [16] by a multiplicative factor of $n^{0.1666}$.

Because r is a constant, our algorithm design focuses on the instances in which $k > r$. Conversely, when $k \leq r$, the optimal solution can be found in $O(n^r)$ time through a brute-force search. Specifically, we develop four algorithms as listed in Table 1. Choosing the better solution from the outputs of the first (resp. last) two algorithms gives an $O(n^{(r-1)/2})$-approximation for kSPM and $\underline{k}$SPM (resp. C$\underline{k}$SPM) on $\mathscr{F}_r$. The solution of C$\underline{k}$SPM is easily expanded to a solution of CkSPM with the same performance guarantee.

Problem	**Approximation ratio**	**Runtime**	**Algorithm**
kSPM ($\underline{k}$SPM)	$O\left(n^{r-1}/k^{r-1}\right)$	$O(n^2)$	Alg. 1 (Greedy Peeling)
kSPM ($\underline{k}$SPM)	$O(k^{r-1})$	$O(n^{r+1})$	Alg. 2 (Batch-greedy Augmenting)
C$\underline{k}$SPM	$O(k^{r-1})$	$O(mn^{r+1})$	Alg. 3 (Star Exploration)
C$\underline{k}$SPM	$O\left(n^{r-1}/k^{r-1}\right)$	$O(mn^2)$	Alg. 4 (Removing-Attaching-Merging)

Table 1: Approximation algorithms for kSPM, $\underline{k}$SPM and C$\underline{k}$SPM problems on $\mathscr{F}_r$, where $k > r$

The algorithms listed in Table 1 differ in their input requirements. Algorithm 1 requires the least information, needing only an oracle for the input function f. Algorithm 2 requires an oracle for f and the value of r. In contrast, Algorithms 3 and 4 are the most demanding, requiring the detailed decomposition information $(V_i, f_i)_{i=1}^m$ of the function $f \in \mathscr{F}_r$, including oracles for each component function f_i, $i = 1, \ldots, m$.

We design Algorithms 1 and 2 not only to solve the kSPM problem but also to serve as subroutines for Algorithms 3 and 4, respectively, to aid in solving the C$\underline{k}$SPM problem.

The approximation ratios of our algorithms for C$\underline{k}$SPM are actually evaluated against the optimum objective value of the $\underline{k}$SPM problem (see Corollary 3.12). The stronger guarantees imply that the optimal solution without the connectivity constraint can be approximated by a connected solution within the stated ratio, though at the expense of a runtime increase by a factor of m.

Algorithms for kSPM and $\underline{k}$SPM. We adopt two complementary strategies to construct a k-subset that approximates the optimal solution of the kSPM problem on $\mathscr{F}_r$. The first method, *Greedy Peeling* (Algorithm 1), starts with the entire set of elements and iteratively removes the one with the smallest marginal contribution until a set of size k remains. This approach yields an approximation ratio of $O(n^{r-1}/k^{r-1})$ (Theorem 2.3).

In contrast, the second method, *Batch-greedy Augmenting* (Algorithm 2), builds a solution from an empty set. Instead of adding a single element one by one, our algorithm employs a batch-greedy strategy, adding an r-subset that provides the maximum increase in function value at each step. This algorithm achieves an $O(k^{r-1})$ approximation (Theorem 2.5). Note that the single-element-greedy augmenting by Bai and Bilmes [7] does not work well even for monotone 2-decomposable supermodular functions, as it suffers from an arbitrarily large approximation ratio independent of $n \geq 4$ (see Appendix A).

By combining these two algorithms and selecting the better of their outputs, we establish an overall approximation ratio of $O(n^{(r-1)/2})$ for the kSPM ($\underline{k}$SPM) problem on $\mathscr{F}_r$ (Theorem 2.6). Note that for the densest k-subhypergraph (DkSH) problem, the current best approximation ratios are $O(n^{1/4+\epsilon})$ for ordinary graphs ($r = 2$) and $O(n^{0.697831+\epsilon})$ for 3-uniform hypergraphs (a special case of $r = 3$). While our algorithm's approximation ratios do not surpass these when

applied to these problems, its key advantage lies in its simplicity and its capability to handle all hypergraphs with rank r.

Our work on greedy peeling for kSPM generalizes the current best $O(n/k)$-approximation ratio (with respect to n and k together) for the densest k-subgraph (DkS) problem [6, 34], which constitutes a special case of the kSPM with $r = 2$. Notably, without the monotonicity assumption on the underlying supermodular function, a slight modification of the greedy peeling provides a $\underline{k}$-subset that is an $O(n^{r-1}/k^{r-1})$-approximation for the $\underline{k}$SPM problem (Corollary 4.1).

Algorithms for C$\underline{k}$SPM. At a high level, the algorithms work by first generating a candidate pool of polynomially many connected $\underline{k}$-subsets; second, identifying and outputting the subset with the maximum value from this pool.

The second-tier strategy of the algorithm design employs a "break-and-build" approach. "Breaking" means fragmenting graph G into "densely" connected components by eliminating elements with "low" utilities. The larger resulting components, which can deliver adequate value, become candidate $\underline{k}$-subsets added to the pool. "Building" operates in two ways: first, by grouping and merging smaller components scattered during the breaking phase to restore connectivity; second, by proactively generating candidate $\underline{k}$-subsets within connected sets using attachment methods. Specifically, to address the additional connectivity constraint, we use the idea of attaching "valuable" elements to a connected subset S (referred to as the core). Based on the core and the input function f, we define a new supermodular function $f^S \in \mathscr{F}_{r-1}$ and construct an accompanying set T that holds potential elements for attachment. All elements $u \in T$ with positive marginal values $f^S(u|T \setminus \{u\})$ are attached to the core to form a candidate $\underline{k}$-subset, where the positive marginals ensure not only the connectivity of the resulting $\underline{k}$-subset (Lemma 3.2) but also, to some extent, its quality.

The third level of our algorithm design is the operational layer, which comprises two complementary algorithms. The first method, *Star Exploration* (Algorithm 3), systematically explores star structures centered at each element (vertex) $v \in V$. For each core $\{v\}$, batch-greedy augmenting (developed in Section 2.2) is used to construct $\{v\}$'s accompanying set T. Then, from T, promising "leaves" (elements with positive marginals) are selected and attached to the center v, forming a star-shaped connected $\underline{k}$-subset, which is added to the candidate pool. The best "star" in the pool yields an approximation ratio of $O(k^{r-1})$ for the C$\underline{k}$SPM problem (Theorem 3.3).

The second method, *Removing-Attaching-Merging* (Algorithm 4), is more intricate. This algorithm operates in stages, beginning with a *removing* procedure (Procedure 2) that deletes elements with low marginal contributions (relative to density) and possibly other elements to produce a dense $\overline{k}$-subset R. The subsequent stage depends on the connectedness of R.

- Either R is connected: An *attaching* procedure (Procedure 1) constructs a connected $\underline{k}$-subset of R as a candidate subset: attaching elements (referred to as petals) to a connected $\lceil (r-1)k/r \rceil$-subset C (the core). The petal selections are guided by function f^C and C's accompanying set T, which is constructed using greedy peeling (presented in Section 2.1).
- Or R is disconnected and contains no connected $\overline{k}$-subset: All maximal connected $\overline{k/3}$-subsets of R are added to the candidate pool; and a *merging* procedure (Procedure 3) groups and connects the remaining maximal connected subsets of R (each with cardinality smaller than $k/3$) into an $O(n/k)$ number of larger candidate connected $\underline{k}$-subsets. The merging process constitutes the most technical part of our algorithm, where a spanning tree in a contracted graph is pruned step by step to show how these $\underline{(k/3-1)}$-subsets are merged under the principle that "closed" subsets have a higher chance of being merged.

By selecting the best connected $\underline{k}$-subset from the candidate pool, this composite algorithm provides an $O(n^{r-1}/k^{r-1})$-approximation for the C$\underline{k}$SPM problem (Theorem 3.9).

Combining Algorithms 3 and 4 (selecting the better of their outputs), we establish an overall approximation ratio of $O(n^{(r-1)/2})$ for the C$\underline{k}$SPM problem on $\mathscr{F}_r$ (Theorem 3.11).

The remainder of this paper is organized as follows. Sections 2 and 3 present approximation algorithms for kSPM and C$\underline{k}$SPM problems, respectively. Section 4 summarizes the paper and discusses possible directions for future research.

2 Approximation under cardinality constraint

For any positive integer h, the symbol $[h]$ denotes the set of all positive integers at most h. Given a nonnegative supermodular function $f : 2^V \to \mathbb{R}_+$ and an integer $k \in [n]$, we are concerned with the kSPM problem for finding a k-subset $S \subseteq V$ that maximizes $f(S)$. By iterative greedy peelings starting from V (see Section 2.1) and iterative batch-greedy augmentations starting from $\emptyset$ (see Section 2.2), we derive a k-subset that is an $O(\min\{n^{r-1}/k^{r-1}, k^{r-1}\})$-approximation for the kSPM problem on $\mathscr{F}_r$. To avoid discussing trivial cases, henceforth we assume $k \in [n-1]$.

Decomposition parameters. Before proceeding with the algorithmic details, we discuss some basic properties of supermodular functions, which are useful in analyzing our algorithms. For any $S \subseteq V$, any normalized supermodular function f satisfies the well-known property that

$$\sum_{v \in S} f(v|S \setminus \{v\}) \geq f(S), \tag{1}$$

which follows from a telescoping sum. Chekuri et al. [14] defined the following parameter that reflects f's quality in terms of modularity and decomposability:

$$c_f := \max_{S \subseteq V} \frac{\sum_{v \in S} f(v|S \setminus \{v\})}{f(S)},$$

where, for notational convenience, when $f(S) = 0$, it is assumed that $\frac{\sum_{v \in S} f(v|S \setminus \{v\})}{f(S)} = 1$. It is straightforward that

$$\min_{v \in S} f(v|S \setminus \{v\}) \leq \frac{\sum_{v \in S} f(v|S \setminus \{v\})}{|S|} \leq c_f \frac{f(S)}{|S|}, \text{ for all } S \subseteq V. \tag{2}$$

Moreover, c_f serves as a lower bound for the decomposition property of the function f.

Proposition 2.1 (Chekuri et al. [14])**.** *For any nonnegative r-decomposable supermodular function $f : 2^V \to \mathbb{R}_+$, it holds that $c_f \leq r$, and therefore $\sum_{v \in S} f(v|S \setminus \{v\}) \leq r \cdot f(S)$ for all $S \subseteq V$.*

Normalization. A function f is *normalized* if $f(\emptyset) = 0$. Any function f can be readily normalized by subtracting the constant $f(\emptyset)$ from its value for every subset. This normalization is a convenient preprocessing step that does not negatively impact our analysis. As shown in Appendix B, the process does not decrease the approximation ratio for the supermodular maximization under consideration. Furthermore, it preserves essential properties of the function, including non-negativity, monotonicity, supermodularity (and r-decomposability). Henceforth, we assume that all such functions are normalized, even if not explicitly stated.

2.1 Greedy peeling

Using a similar idea to Kortsarz and Peleg [34] and Asahiro et al. [6] for finding a dense subgraph, we give the following algorithm for kSPM, which repeatedly removes the element with the currently smallest marginal contribution.

Algorithm 1: Greedy peeling for k-subsets
Input: Supermodular function $f : 2^V \to \mathbb{R}_+$ and integer $k \in [n-1]$
$S_n \leftarrow V$;
for $i \leftarrow n$ **downto** $k+1$ **do**
  $v_i \leftarrow \operatorname{argmin}_{v \in S_i}\{f(v|S_i \setminus \{v\})\}$;
  $S_{i-1} \leftarrow S_i \setminus \{v_i\}$;
Output $\textsc{Alg1}(f, k) := S_k$.

Note that the execution of Algorithm 1 does not make any assumption on the decomposability of the underlying supermodular function. Even the value of the parameter r does not need to be known.

Lemma 2.2. *If f is r-decomposable and $r < k$, then $f(S_i)/f(S_j) \geq \binom{i}{r}/\binom{j}{r} \geq \frac{r!}{r^r} \cdot \frac{i^r}{j^r}$ for all i, j with $k \leq i \leq j \leq n$.*

Proof. For every integer $h \in [k+1, n]$, we have

$$f(S_{h-1}) = f(S_h) - \min_{v \in S_h} f(v|S_h \setminus \{v\}) \geq f(S_h) - c_f \frac{f(S_h)}{h} \geq f(S_h) - r\frac{f(S_h)}{h} = \frac{h-r}{h} f(S_h),$$

where the first and second inequalities follow from (2) and Proposition 2.1, respectively. Applying the inequality $f(S_{h-1}) \geq \frac{h-r}{h} f(S_h)$ repeatedly yields

$$\begin{aligned} f(S_i) &\geq \frac{(j-r) \cdot (j-r-1) \cdots (i-r+1)}{j \cdot (j-1) \cdots (i+1)} \cdot f(S_j) \\ &= \frac{(j-r)!/(i-r)!}{j!/i!} \cdot f(S_j) \\ &= \frac{i!/(i-r)!}{j!/(j-r)!} \cdot f(S_j) \\ &= \frac{\binom{i}{r}}{\binom{j}{r}} \cdot f(S_j) \end{aligned}$$

On the other hand, it follows from $i > r$ that $\frac{i(i-1)\cdots(i-r+1)}{r!} \geq (\frac{i}{r})^r$ and $\binom{i}{r}/\binom{j}{r} \geq \frac{i \cdot (i-1) \cdots (i-r+1)}{j^r} \geq \frac{r!}{r^r} \cdot \frac{i^r}{j^r}$, proving the lemma. □

Theorem 2.3. *Algorithm 1 runs in $O(n^2)$ time, and achieves an approximation ratio of $O(n^{r-1}/k^{r-1})$ for the kSPM problem on $\mathscr{F}_r$, provided $k > r$.*

Proof. The running time is obvious. To prove the approximation ratio, let $f \in \mathscr{F}_r$ and S_{opt} be an optimal solution of the problem and $\mathrm{OPT} := f(S_{opt})$. We construct a sequence of shrinking sets $S'_k, S'_{k-1}, \ldots, S'_0$ in this order. Initially set $S'_k := S_{opt}$. Iteratively, for each $h = k, k-1, \ldots, 1$, given S'_h, if there is $v'_h \in S'_h$ with $f(v'_h|S'_h \setminus \{v'_h\}) \leq \mathrm{OPT}/(2k)$, then let $S'_{h-1} := S'_h \setminus \{v'_h\}$; otherwise, let $S'_{h-1} := S'_h$. So we have $S_{opt} = S'_k \supseteq S'_{k-1} \supseteq \cdots \supseteq S'_0$, $f(S'_0) \geq \mathrm{OPT} - k \cdot \mathrm{OPT}/(2k) = \mathrm{OPT}/2$ and $f(v|S'_0 \setminus \{v\}) > \mathrm{OPT}/(2k)$ for each $v \in S'_0$.

First, we consider the case that $f(v_i|S_i \setminus \{v_i\}) \leq \mathrm{OPT}/(2k)$ for each $i \in [k+1, n]$. We claim that $S'_0 \subseteq S_k$. Otherwise, there exists $i \in [k+1, n]$ such that $S'_0 \subseteq S_i$ and $v_i \in S'_0$. However, since f is supermodular, $f(v_i|S_i \setminus \{v_i\}) \geq f(v_i|S'_0 \setminus \{v_i\}) > \mathrm{OPT}/(2k)$, a contradiction. Moreover, since f is monotone, we have

$$\frac{\mathrm{OPT}}{f(S_k)} \leq \frac{\mathrm{OPT}}{f(S'_0)} \leq 2.$$

Now, the remaining case is that there exists $t \in [k+1, n]$ such that $f(v|S_t \setminus \{v\}) \geq \mathrm{OPT}/(2k)$ for each $v \in S_t$. Then by Lemma 2.2 and Proposition 2.1, we have

$$\begin{aligned} f(S_k) &\geq \frac{r!}{r^r} \cdot \frac{k^r}{t^r} \cdot f(S_t) \\ &\geq \frac{r!}{r^r} \cdot \frac{k^r}{t^r} \cdot \frac{\sum_{v \in S_t} f(v|S_t \setminus \{v\})}{r} \\ &\geq \frac{r!}{r^r} \cdot \frac{k^r}{t^r} \cdot \frac{t \cdot \mathrm{OPT}/(2k)}{r} \end{aligned}$$

and

$$\frac{\mathrm{OPT}}{f(S_k)} \leq \frac{r^r \cdot 2r}{r!} \cdot \frac{t^{r-1}}{k^{r-1}} = O(n^{r-1}/k^{r-1}),$$

establishing the theorem. □

2.2 Batch-greedy augmenting

Unlike its high efficiency in maximizing submodular functions, the classical greedy augmenting algorithm–starting from the empty set and greedily adding elements one by one–typically performs poorly for supermodular maximization (see Appendix A for examples). We show, however,

that for r-decomposable functions, a batch-greedy variant that adds r elements at a time (while respecting the cardinality constraint) achieves a provably good approximation.

Algorithm 2: Batch-greedy augmenting for k-subsets

Input: Supermodular function $f : 2^V \to \mathbb{R}_+$, integers r and k such that $1 \le r \le k \le n-1$

$T_0 \leftarrow \emptyset$;
$t \leftarrow \lfloor k/r \rfloor$;
for $i \leftarrow 1$ **to** t **do**
    $\mathcal{R}_i \leftarrow$ the set of r-subsets of $V \setminus T_{i-1}$;
    $S_i \leftarrow$ an r-subset in $\operatorname{argmax}_{S \in \mathcal{R}_i} f(T_{h-1} \cup S)$;
    $T_i \leftarrow T_{i-1} \cup S_i$;
Output ALG2$(f, r, k) :=$ any k-subset of V that contains T_t

Lemma 2.4. *Let $U_1, U_2, \ldots, U_t$ be any t pairwise disjoint r-subsets of V. If f is monotone, then $f(T_t) \ge \frac{1}{2} \sum_{h=1}^{t} f(U_h)$.*

Proof. Suppose without loss of generality that $f(U_1) \ge \cdots \ge f(U_t)$. We prove $f(T_i) \ge \frac{1}{2} \sum_{h=1}^{i} f(U_h)$ by induction on $i = 1, 2, \ldots, t$. The inequality trivially holds when $i = 1$ because $U_1 \in \mathcal{R}_1$ and $f(T_1) = f(S_1) \ge f(U_1)$. Proceeding inductively, we assume that $i \ge 2$ and $f(T_{i-1}) \ge \frac{1}{2} \sum_{h=1}^{i-1} f(U_h)$.

First we consider the case in which $f(U_g \cap T_{i-1}) \le f(U_g)/2$ for some $g \le i$. Clearly, there exists $S \in \mathcal{R}_i$ such that $U_g \subseteq T_{i-1} \cup S$. The supermodularity of f implies $f(U_g \cup S) + f(T_{i-1}) \le f(T_{i-1} \cup S) + f(T_{i-1} \cap U_g)$. In turn, the definition of T_i gives $f(T_i) - f(T_{i-1}) \ge f(T_{i-1} \cup S) - f(T_{i-1}) \ge f(U_g \cup S) - f(T_{i-1} \cap U_g)$. Since f is monotone, we have

$$f(T_i) - f(T_{i-1}) \ge f(U_g) - f(T_{i-1} \cap U_g) \ge f(U_g) - f(U_g)/2 = f(U_g)/2.$$

By the induction hypothesis and $f(U_g) \ge f(U_i)$, we obtain

$$f(T_i) = f(T_{i-1}) + (f(T_i) - f(T_{i-1})) \ge \frac{\sum_{h=1}^{i-1} f(U_h)}{2} + \frac{f(U_g)}{2} \ge \frac{\sum_{h=1}^{i} f(U_h)}{2}.$$

It remains to consider the case in which $f(U_h \cap T_{i-1}) \ge f(U_h)/2$ for all $h \le i$. Since $U_1, \ldots, U_i$ are mutually disjoint and $f(\emptyset) = 0$, it follows from the supermodularity and monotonicity of f that $\sum_{h=1}^{i} f(U_h \cap T_{i-1}) \le f((\cup_{h=1}^{i} U_h) \cap T_{i-1}) \le f(T_{i-1})$. Furthermore,

$$f(T_i) \ge f(T_{i-1}) \ge \sum_{h=1}^{i} f(U_h \cap T_{i-1}) \ge \frac{\sum_{h=1}^{i} f(U_h)}{2},$$

which completes the proof of the lemma. □

Theorem 2.5. *Algorithm 2 runs in $O(n^{r+1})$ time, and achieves an approximation ratio of $O(k^{r-1})$ for the kSPM problem on $\mathscr{F}_r$, provided $k > r$.*

Proof. Note that each $\mathcal{R}_i$ in the algorithm is of size $O(n^r)$ and can be constructed in $O(n^r)$ time. It follows that the algorithm runs in $O(n^{r+1})$ time.

To prove the approximation ratio, suppose that the r-decomposition of $f \in \mathscr{F}_r$ is $(V_i, f_i)_{i=1}^m$. (This decomposition is not an input to the algorithm; it is only used for the proof.) Let S_{opt} be an optimal solution of the problem and OPT $:= f(S_{opt})$. For each $i = 1, \ldots, m$, because $|V_i| \le r$, we see that $V_i \cap S_{opt}$ is an $\underline{r}$-subset of both V_i and S_{opt}, and therefore the intersection of V_i and some r-subset of S_{opt}. Let $\mathcal{R}$ consist of all r-subsets of S_{opt}. It follows from f's decomposability and f_i's nonnegativity that

$$\mathrm{OPT} = \sum_{i=1}^{m} f_i(V_i \cap S_{opt}) \le \sum_{i=1}^{m} \sum_{S \in \mathcal{R}} f_i(V_i \cap S) = \sum_{S \in \mathcal{R}} \sum_{i=1}^{m} f_i(V_i \cap S) = \sum_{S \in \mathcal{R}} f(S).$$

Let $\mathcal{U}$ be the set of vectors $(U_1, \ldots, U_t)$ with $U_1, \ldots, U_t$ being mutually disjoint r-subsets of S_{opt}. It is instant from Lemma 2.4 that $\sum_{(U_1,\ldots,U_t) \in \mathcal{U}} \sum_{i=1}^{t} f(U_i) \le 2f(T_t) \cdot |\mathcal{U}|$. On the other hand, by symmetry, the number of appearances of an r-subset in $\mathcal{R}$ as an element of a vector in $\mathcal{U}$ is

the same value $t|\mathcal{U}|/|\mathcal{R}|$, which gives $\sum_{(U_1,\dots,U_t)\in\mathcal{U}} \sum_{i=1}^{t} f(U_i) = (t|\mathcal{U}|/|\mathcal{R}|)\cdot\sum_{S\in\mathcal{R}} f(S)$. Hence $\sum_{S\in\mathcal{R}} f(S) \le 2f(T_t)|\mathcal{R}|/t = 2f(T_t)\binom{k}{r}/t \le 2f(T_t)k^r/t$. Since $t = \lfloor k/r \rfloor \ge (k-r+1)/r$, we have

$$\mathrm{OPT} \le \sum_{S\in\mathcal{R}} f(S) \le 2r \cdot \frac{k^r}{k-r+1} \cdot f(T_t) = O(k^{r-1}) \cdot f(T_t).$$

□

2.3 Combining

The combination of Theorems 2.3 and 2.5 gives the following main result of this section. Taking the better solution output by Algorithms 1 and 2 (when $k > r$), and using brute-force search to find the optimum (in case of $k \le r$), we can guarantee an approximation ratio of $O(\min\{k^{r-1}, n^{r-1}/k^{r-1}\})$, where $\min\{k^{r-1}, n^{r-1}/k^{r-1}\} \le n^{(r-1)/2}$.

Theorem 2.6. *There is an $O(n^{(r-1)/2})$-approximation algorithm for the kSPM and $\underline{k}$SPM problems on $\mathscr{F}_r$, which runs in $O(n^{r+1})$ time.*

3 Approximation under cardinality and connectivity constraints

In this section, we design two approximation algorithms (Algorithm 3 and 4) for the connected $\underline{k}$-subset supermodular maximization (C$\underline{k}$SPM) problem on $\mathscr{F}_r$, achieving approximation ratios $O(k^{r-1})$ and $O(n^{r-1}/k^{r-1})$, respectively. Combining these two algorithms, we derive an $O(n^{(r-1)/2})$-approximation for the C$\underline{k}$SPM problem. These results extend to the CkSPM problem due to their "indirect equivalence", as discussed in Section 1.

Given function $f : 2^V \to \mathbb{R}_+$ in $\mathscr{F}_r$, without loss of generality, we assume f is normalized (see Appendix B). Unlike the kSPM ($\underline{k}$SPM) counterpart, which only require the value of r during our algorithmic solving process (i.e., the greedy augmenting subroutine), the C$\underline{k}$SPM setting explicitly provides the r-decomposition $(V_i, f_i)_{i=1}^m$ of f as an input, where every V_i is an $\underline{r}$-subset of V, every $f_i : 2^{V_i} \to \mathbb{R}_+$ is supermodular, and the following *decomposition identities* hold:

$$f(S) = \textstyle\sum_{i=1}^m f_i(S \cap V_i) \text{ for each } S \subseteq V.$$

For brevity, we denote the input of the C$\underline{k}$SPM problem by $f \in \mathscr{F}_r^V$, which specifies the ground set V and implicitly provides the corresponding r-decomposition.

Underlying graph. This decomposition is used to define the connectivity constraint of the C$\underline{k}$SPM problem. Specifically, $(V_i)_{i=1}^m$ yields an undirected graph $G = (V, E)$ with vertex set V and edge set $E = \cup_{i=1}^m \{uv : \{u, v\} \subseteq V_i, u \ne v\}$. The objective of the C$\underline{k}$SPM is to find a connected $\underline{k}$-subset $S \subseteq V$ such that $f(S)$ is maximized, where "connected" means that $G[S]$, the subgraph of G induced by S, is connected.

We only need to consider the case where V *is connected*. This, in particular, implies that $r \ge 2$ and there is a k-subset that is optimal for the C$\underline{k}$SPM problem. If the graph G is not connected, we can reduce the problem by considering each connected component separately: solving C$\underline{l}$SPM for the vertex set U of each connected component of G, where $l = \min\{k, |U|\}$, and selecting the best solution as the overall output for the C$\underline{k}$SPM on V. Note that this reduction does not change the approximation ratio.

Given that the ground set V of the function f coincides with the vertex set of the graph G, we shall *use the terms "element" and "vertex" interchangeably.*

Decomposition constituents. Notice from $0 = f(\emptyset) = \sum_{i=1}^m f_i(\emptyset)$ that all $f_i : 2^{V_i} \to \mathbb{R}_+$ are normalized. So $f_i(v|\emptyset) = f_i(v) \ge 0$ holds for all $v \in V_i$. In turn, f_i's supermodularity implies that it is monotone, since $f_i(v|S) \ge f_i(v|\emptyset) \ge 0$ for any $S \subset V_i$ and any $v \in V_i \setminus S$. Thus

- for each $i \in [m]$, $|V_i| \le r$ and $f_i : 2^{V_i} \to \mathbb{R}_+$ is nonnegative, supermodular, normalized and monotone.

By f's decomposability, we have $f(S) = \sum_{i=1}^m f_i(S \cap V_i) = f(S \cap (\cup_{i=1}^m V_i))$ for every $S \subseteq V$. Therefore, for notational convenience, we may assume without loss of generality that

- $V = \cup_{i=1}^m V_i$.

Algorithm overview. Before presenting the the technical details, we briefly outline our approach for finding a high-valued connected $\underline{k}$-subset. In Section 3.1, we construct candidate $\underline{k}$-subsets by "attaching" elements to a common connected core, which takes one of two forms: either a single vertex serving as a star center (Section 3.1.1) or a connected $\lceil (r-1)k/r \rceil$-subset functioning as a sunflower seed disc (Section 3.1.2). In Section 3.2, we remove so-called removable elements from V to "lift" both the density and marginal values of elements, and we further drop redundant elements whenever an entire maximal connected $\overline{k}$-subset survives. This process always maintains a set with at least k elements, but its connectivity may be compromised. If the resulting set R remains connected, then the sunflower attachment procedure, developed in Section 3.1.2, applies to construct a $\underline{k}$-subset of R, yielding an $O(n^{r-1}/k^{r-1})$-approximation for the C$\underline{k}$SPM problem (see Lemma 3.7). If R is not connected and contains no connected $\overline{k}$-subsets, we proceed to merging step in Section 3.3. Here, we merge "small" maximal connected subsets of R to form larger connected $\underline{k}$-subsets of V. We then select the subset with the highest value from among these newly formed subsets and existing "big" maximal connected subsets of R. Such a maximum-valued subset gives an $O(n/k)$-approximation of the C$\underline{k}$SPM problem (see Lemma 3.8). Finally, in Section 3.4, we integrate the removing, attaching, and merging procedures to derive an $O(n^{r-1}/k^{r-1})$-approximation in $O(mn^2)$ time. Combined with the $O(k^{r-1})$-approximation from the best star attachments (Section 3.1.1), this yields an overall $O(n^{(r-1)/2})$-approximation for the C$\underline{k}$SPM problem (see Theorem 3.11).

3.1 Concentric attachments

We say that two disjoint subsets of V are *connected to* each other if there exists a vertex in each of these two subsets such that the two vertices are neighbors in G. To ensure the connectivity of the $\underline{k}$-subset we construct, the most straightforward idea is to attach some elements to a connected subset at hand (referred to as the *core*), where each element is connected to this core. In this way, the resulting $\underline{k}$-subset will be connected. For this purpose, we need to define a function f^S for the core S, and determine whether an element is connected to S based on its marginal value with respect to some other set under f^S.

For each nonempty $S \subseteq V$, let I_S denote the set of indices $i \in [m]$ with $V_i \cap S \neq \emptyset$, and let function $f^S : 2^{V \setminus S} \to \mathbb{R}$ be defined by

$$f^S(T) := \sum_{i \in I_S} \big(f_i((S \cup T) \cap V_i) - f_i(S \cap V_i) \big) \text{ for all } T \subseteq V \setminus S.$$

It is clear that f^S is nonnegative and monotone. Note from f's decomposition identities and f_i's normality and nonnegativity that

$$f^S(T) = \left(\sum_{i \in I_S} f_i((S \cup T) \cap V_i) \right) - f(S) \leq f(S \cup T) - f(S) \text{ for all } T \subseteq V \setminus S. \tag{3}$$

The leftmost equation in (3) and the supermodularity of f_i imply that f^S is supermodular. For each $i \in I_S$, let $f_i^S : 2^{V_i \setminus S} \to \mathbb{R}$ be defined by $f_i^S(T) := f_i((S \cup T) \cap V_i) - f_i(S \cap V_i)$ for all $T \subseteq V_i \setminus S$. Then we obtain the following decomposability for f^S.

Lemma 3.1. *For any nonempty $S \subseteq V$, the function $f^S : 2^{V \setminus S} \to \mathbb{R}_+$ is nonnegative, monotone, and supermodular. Moreover, it admits an $(r-1)$-decomposition $(V_i \setminus S, f_i^S)_{i \in I_S}$, where functions f_i^S, $i \in I_S$, are nonnegative, monotone and supermodular.* □

Lemma 3.2. *For any $T \subseteq V \setminus S$ and $u \in V \setminus S$, if $f^S(u|T) > 0$, then $\{u\}$ is connected to S.*

Proof. By the leftmost equation in (3), we deduce from $f^S(u|T) > 0$ that there must exist some $i \in I_S$ such that $(S \cup T \cup \{u\}) \cap V_i$ *properly* contains $(S \cup T) \cap V_i$, which particularly implies that $u \in V_i \setminus S$. By definition, $I_S \neq \emptyset$ gives $S \cap V_i \neq \emptyset$. In turn, the construction of G says that vertex u is a neighbor of every vertex in $S \cap V_i \neq \emptyset$. □

The lemma provides us a preliminary approach to ensure connectivity: starting from some connected subset S (the core), seeking an appropriate *accompanying set* $T \subseteq V \setminus S$, and picking the elements u with positive marginal values $f^S(u|T \setminus \{u\})$, and attaching all these u to core S to form a connected $\underline{k}$-subset that possesses a large value under f. In the process, the selection

of core S and the construction of accompanying set T are two crucial factors, which jointly determine the *attachments* (those u attached).

In the next two subsections, we consider two cases for the core S. In the first case (Section 3.1.1), we start with a 1-subset (trivially connected) as the core (center) and expand it to a "star" by attaching "leaves". In the second case (Section 3.1.2), we take a connected $\lceil (r-1)k/r \rceil$-subset as the core (seed disc) to form a "sunflower" by attaching "petals". The accompanying set T is then constructed using the batch-greedy augmentation procedure (introduced in Section 2.2) for the star and the greedy peeling procedure (presented in Section 2.1) for the sunflower, with the leaf and petal attachments selected according to Lemma 3.2.

3.1.1 Star exploration

Using an idea similar to that in Chen et al. [16], we derive the following algorithm for C$\underline{k}$SPM, which greedily explores stars centered at all vertices v of graph G, and outputs the best one as an $O(k^{r-1})$-approximate solution (Theorem 3.3).

Algorithm 3: Star exploration for connected $\underline{k}$-subsets

Input: $f \in \mathscr{F}_r^V$ and $k \in [n-1]$

1 **for** $v \in V$ **do**

2 $\quad T^v \leftarrow \textsc{Alg2}(f^{\{v\}}, r-1, k-1)$;

3 $\quad L^v \leftarrow \{u \in T^v : f^{\{v\}}(u|T^v \setminus \{u\}) > 0\}$;

4 $\quad S^v \leftarrow L^v \cup \{v\}$;

5 **Output** $\textsc{Alg3}(f,k) :=$ any $\underline{k}$-subset in $\operatorname{argmax}_{S^v}\{f(S^v)\}$

The greedy nature here is reflected in the fact that the set T (i.e., T^v in the algorithm) accompanying the core $\{v\}$ is a $(k-1)$-subset constructed in Step 2 using the greedy augmenting (Algorithm 2), specifically for the $(r-1)$-decomposable supermodular function $f^{\{v\}}$. In Step 3, we select all $u \in T^v$ that make positive marginal contribution to $T^v \setminus \{u\}$, where the $(k-1)$-subset L^v holds the vertices selected. By Lemma 3.2, each vertex in L^v is connected to $\{v\}$. Therefore, each $\underline{k}$-subset S^v generated in Step 4 constitutes the vertex set of a star centered at v, which particularly shows that the $\underline{k}$-subset output by Algorithm 3 is connected.

Theorem 3.3. *Algorithm 3 runs in $O(mn^{r+1})$ time and achieves an approximation ratio of $O(k^{r-1})$ for the C$\underline{k}$SPM problem on $\mathscr{F}_r$, provided $k > r$.*

Proof. When calling the subroutine of Algorithm 2, it necessitates invoking the oracle for $f^{\{v\}}$ $O(n^{r+1})$ times (recalling Theorem 2.5). Furthermore, according to the definition of $f^{\{v\}}$, one run of its oracle requires $O(m)$ calls to the oracles of the f_i's. Consequently, the overall running time is upper bounded by $O(mn^{r+1})$.

Let S^*_{opt} be an optimal solution of the $\underline{k}$SPM and $\mathrm{OPT}^* := f(S^*_{opt})$. Then OPT^* is an upper bound of the optimal objective value of the C$\underline{k}$SPM, and

$$\sum_{v \in S^*_{opt}} f(v|S^*_{opt} \setminus \{v\}) \geq \mathrm{OPT}^*$$

by f's supermodularity (recall (1)). Notice from Theorem 2.5 that there is a universal constant Φ such that $f^{\{v\}}(S^*_{opt} \setminus \{v\}) \leq \Phi k^{r-2} \cdot f^{\{v\}}(T^v)$ holds for all $v \in S^*_{opt}$. For every $v \in S^*_{opt}$, since $f^{\{v\}}$ is supermodular (by Lemma 3.1), it is straightforward from its increasing marginal returns that $f^{\{v\}}(L^v) = f^{\{v\}}(T^v)$. Recall from (3) that $f(S^v) \geq f(v) + f^{\{v\}}(L^v)$. In turn, f's nonnegativity gives

$$f(S^v) \geq f(v) + f^{\{v\}}(T^v) \geq \frac{f(v) + f^{\{v\}}(S^*_{opt} \setminus \{v\})}{\Phi k^{r-2}} \text{ for every } v \in S^*_{opt}.$$

Furthermore, since f_i's are normalized, $f(v) = \sum_{i \in [m]} f_i(\{v\} \cap V_i) = \sum_{i \in [m]: v \in V_i} f_i(v)$. By the definition of $f^{\{v\}}$ and the nonnegativity of f_i's, we have

$$f(v) + f^{\{v\}}(S^*_{opt} \setminus \{v\}) = \sum_{i \in [m]: v \in V_i} f_i(S^*_{opt} \cap V_i)$$

$$= \sum_{i=1}^{m} f_i(S^*_{opt} \cap V_i) - \sum_{i\in[m]:v\notin V_i} f_i(S^*_{opt} \cap V_i)$$
$$\geq \sum_{i=1}^{m} f_i(S^*_{opt} \cap V_i) - \sum_{i=1}^{m} f_i((S^*_{opt} \setminus \{v\}) \cap V_i)$$

It follows from f's decomposition identities that

$$f(v) + f^{\{v\}}(S^*_{opt} \setminus \{v\}) \geq f(S^*_{opt}) - f(S^*_{opt} \setminus \{v\}) = f(v|S^*_{opt} \setminus \{v\}) \text{ for every } v \in S^*_{opt}.$$

Therefore, the $\underline{k}$-subset output by Algorithm 3 has value at least

$$\max_{v\in V} f(S^v) \geq \frac{\sum_{v\in S^*_{opt}} f(S^v)}{|S^*_{opt}|} \geq \frac{\sum_{v\in S^*_{opt}} f(v|S^*_{opt} \setminus \{v\})}{k \cdot \Phi k^{r-2}} \geq \frac{\mathrm{OPT}^*}{\Phi k^{r-1}},$$

proving the $O(k^{r-1})$ approximation ratio. □

Note that, in the above proof, the algorithm's performance is measured against the optimal solution S^*_{opt} of the $\underline{k}$SPM problem, which does not impose the connectivity constraint.

Corollary 3.4. *Provided that $k > r$ and V is connected, Algorithm 3 outputs an $O(k^{r-1})$-approximate solution for the $\underline{k}$SPM problem on $\mathscr{F}_r$, which is connected.*

In the star exploration of Algorithm 3, the initially connected set, comprising only a single center vertex, could potentially be too small for some instances. As a result, the leaf attachments might be disproportionately scattered, making it challenging to achieve a good balance between connectivity and the attained objective value. Conversely, if we start with a larger initial connected set C — the seed disc of a sunflower, then the selection of petal attachments according to f^C may have more space to achieve a higher objective value.

3.1.2 Sunflower growth

In this subsection, we take any connected $\lceil (r-1)k/r \rceil$-subset C as the sunflower disc (which, for example, can be constructed in $O(m)$ time by breadth-first search in G). Then, we use greedy peeling (introduced in Section 2) to find the corresponding attachments, as specified in the following algorithm.

Procedure 1: Sunflower growth for connected $\underline{k}$-subsets
Input: $f \in \mathscr{F}_r^V$ and $k \in [n-1]$
1 $C \leftarrow$ a connected $\lceil (r-1)k/r \rceil$-subset of V;
2 $T \leftarrow \textsc{Alg1}(f^C, \lfloor k/r \rfloor)$;
3 $P \leftarrow \{u \in T : f^C(u|T \setminus \{u\}) > 0\}$;
4 **Output** $\textsc{Prc1}(f,k) := C \cup P$.

Although the procedure indeed outputs a feasible solution for C$\underline{k}$SPM (see Lemma 3.5), its approximation ratio against the optimum can be arbitrarily bad in the worst case. On the other hand, its performance could be quantified in terms of a universal lower bound δ on element marginal values and a positive constant c, which are defined as follows:

$$\delta := \min_{v\in V} f(v|V \setminus \{v\}) \text{ and } c := \frac{(r-1)!}{r^r \cdot (2r-2)^{r-1}}.$$

Lemma 3.5. *Procedure 1 runs in $O(mn^2)$ time, and outputs a connected k-subset $S := C \cup P$ of V such that $f(S) \geq c \cdot (k/n)^{r-1} \cdot k\delta$.*

Proof. The runtime is dominated by the computation at Step 2. By Theorem 2.3, it involves $O(n^2)$ calls of f^C's oracle, each of which consists of $O(m)$ calls of f_i's oracles.

Note that f^C is defined on $2^{V\setminus C}$, giving that T is a $\lfloor k/r \rfloor$-subset of $V \setminus C$. For each $u \in P$, since $f^C(u|T \setminus \{u\}) > 0$, by Lemma 3.2, $\{u\}$ is connected to C. It follows from C's connectivity that S is a connected k-subset of V.

It is straightforward that $f^C(T) = f^C(P)$. By Lemma 2.2 and the $(r-1)$-decomposability of f^C (recalling Lemma 3.1), we have

$$\begin{aligned} f^C(T) &\geq \left(\frac{(r-1)!}{(r-1)^{r-1}} \cdot \left\lfloor \frac{k}{r} \right\rfloor^{r-1} \Bigg/ \left(n - \left\lceil \frac{(r-1)k}{r} \right\rceil \right)^{r-1} \right) \cdot f^C(V \setminus C) \\ &\geq \frac{(r-1)!}{(r-1)^{r-1}} \cdot \left(\frac{k}{2r} \right)^{r-1} \cdot \left(\frac{1}{n} \right)^{r-1} f^C(V \setminus C) \\ &= \frac{(r-1)!}{(2r-2)^{r-1}} \cdot \left(\frac{k}{rn} \right)^{r-1} f^C(V \setminus C), \end{aligned}$$

where the second inequality follows from $k/r - \lfloor k/r \rfloor < 1 \leq \lfloor k/r \rfloor$. By f's decomposition identities, we have

$$f(v|V \setminus \{v\}) = f(V) - f(V \setminus \{v\}) = \sum_{i:v \in V_i} f_i(V_i) - f_i(V_i \setminus \{v\}) \leq \sum_{i:v \in V_i} f_i(V_i),$$

and

$$\begin{aligned} f^C(V \setminus C) &= \left(\sum_{i:V_i \cap C \neq \emptyset} f_i(V_i) \right) - f(C) \\ &= \left(\sum_{v \in C} \sum_{i:v \in V_i} \frac{f_i(V_i)}{|V_i \cap C|} \right) - f(C) \\ &\geq \sum_{v \in C} \left(\sum_{i:v \in V_i} \frac{f_i(V_i)}{r-1} - \sum_{i:v \in V_i \subseteq C} \frac{f_i(V_i)}{r(r-1)} \right) - f(C) \\ &\geq \sum_{v \in C} \frac{f(v|V \setminus \{v\})}{r-1} - \sum_{i:V_i \subseteq C} \sum_{v \in V_i} \frac{f_i(V_i)}{r(r-1)} - f(C) \\ &\geq \delta \cdot \frac{(r-1)k}{r} \cdot \frac{1}{r-1} - \sum_{i:V_i \subseteq C} \frac{|V_i| \cdot f_i(C)}{r(r-1)} - f(C) \\ &\geq \frac{\delta k}{r} - \frac{f(C)}{r-1} - f(C) \\ &= \frac{\delta k}{r} - \frac{r f(C)}{r-1}, \end{aligned}$$

where the first equation is implied by f_i's normality, and the third inequality is implied by their monotonicity. Recall from (3) that $f(S) = f(C \cup P) \geq f(C) + f^C(P)$. Thus,

$$\begin{aligned} f(S) &\geq f(C) + f^C(T) \\ &\geq f(C) + \frac{(r-1)!}{(2r-2)^{r-1}} \cdot \left(\frac{k}{rn} \right)^{r-1} \cdot \left(\frac{k\delta}{r} - \frac{r f(C)}{r-1} \right) \\ &= \frac{(r-1)!}{(2r-2)^{r-1}} \cdot \frac{1}{r^r} \left(\frac{k}{n} \right)^{r-1} \cdot k\delta + \left(1 - \frac{r!}{(r-1)^r \cdot (2r)^{r-1}} \cdot \left(\frac{k}{n} \right)^{r-1} \right) f(C) \\ &\geq c \cdot (k/n)^{r-1} \cdot k\delta, \end{aligned}$$

establishing the lemma. □

3.2 Element removal

Recall from Theorem 1.1 that one can find in $O(n^2)$ time a $\overline{k}$-subset $\bar{V}$ of V that is an $(r+1)$-approximation of the densest $\overline{k}$-subset, i.e., $(r+1) \cdot \frac{f(\bar{V})}{|\bar{V}|} \geq \frac{f(S)}{|S|}$ for every $\overline{k}$-subset S of V. Since $f \in \mathscr{F}_r$ and V is connected, the C$\underline{k}$SPM admits a k-subset as an optimal solution. Let OPT denote the optimal objective value of the C$\underline{k}$SPM. Then

$$\frac{f(\bar{V})}{|\bar{V}|} \geq \frac{\mathrm{OPT}}{k(r+1)}. \tag{4}$$

For any $S \subseteq V$, an element $v \in S$ is called *removable* in S if $f(S \setminus \{v\})/(|S|-1) > f(S)/|S|$.

Procedure 2: Element removals from $\overline{k}$-subsets
Input: $f \in \mathscr{F}_r^V$ and $k \in [n-1]$
$T \leftarrow$ a $\overline{k}$-subset $\bar{V}$ of V that satisfies (4);
while $|T| > k$ *and* T *has a removable element* **do**
  $v \leftarrow$ a removable element of T;
  $T \leftarrow T \setminus \{v\}$;
$R \leftarrow T$;
if $|T| > k$ *and* T *contains a connected* $\overline{k}$*-subset* **then**
  $R \leftarrow$ a maximal connected $\overline{k}$-subset of T;
Output $\textsc{Prc2}(f,k) := R$

Henceforth, we reserve symbol R to represent the output of Procedure 2, which takes f, its r-decomposition and k as input.

Moreover, the restriction of f to 2^R, denoted as $f|_{2^R}$, admits an r-decomposition $(V_i \cap R, f_i|_{2^{V_i \cap R}})_{i \in I_R}$, where I_R consists the indices $i \in [m]$ with $V_i \cap R \neq \emptyset$. Thus, we can write $f|_{2^R} \in \mathscr{F}_r^R$ for brevity.

Lemma 3.6. *Procedure 2 runs in $O(n^2)$ time and outputs a $\overline{k}$-subset R of V. One of the following holds:*
(a) The procedure skip Step 7, and $f(R) \geq \mathrm{OPT}/(r+1)$.
(b) The procedure executes Step 7, R is connected, $f(v|R \setminus \{v\}) \geq \mathrm{OPT}/(k(r+1))$ for each $v \in R$ and $f(R) \geq \mathrm{OPT}/(r(r+1))$.

Proof. Clearly, $|R| \geq k$, and the runtime is dominated by the computation of $\bar{V}$, which takes $O(n^2)$ time.

By the definition of removal elements, when the procedure terminates, the final T satisfies $f(T)/|T| \geq f(\bar{V})/|\bar{V}|$. Furthermore inequality (4) gives

$$\frac{f(T)}{|T|} \geq \frac{\mathrm{OPT}}{k(r+1)}.$$

The validity of (a) is straightforward, given that the set R is assigned its final setting at Step 5 of the procedure.

To prove (b), we assume that the procedure executes Step 7. Then the final T, which consists of more than k elements, cannot have any removable element. Thus, by definition, for each $v \in T$, we have $f(T \setminus \{v\}) \leq (|T|-1)f(T)/|T|$, and

$$f(v|T \setminus \{v\}) = f(T) - f(T \setminus \{v\}) \geq f(T) - f(T) \cdot \frac{|T|-1}{|T|} = \frac{f(T)}{|T|} \geq \frac{\mathrm{OPT}}{k(r+1)}.$$

Observe that R induces a connected component in $G[T]$. So $f(v|R \setminus \{v\}) = f(v|T \setminus \{v\}) \geq \mathrm{OPT}/(k(r+1))$ for each $v \in R$. Besides, since f is r-decomposable, it follows from Proposition 2.1 that

$$rf(R) \geq \sum_{v \in R} f(v|R \setminus \{v\}) \geq \sum_{v \in R} \frac{\mathrm{OPT}}{k(r+1)} = |R| \cdot \frac{\mathrm{OPT}}{k(r+1)} \geq \frac{\mathrm{OPT}}{r+1}.$$

It leads to $f(R) \geq \mathrm{OPT}/(r(r+1))$ as desired. □

Lemma 3.7. *If $R = \textsc{Prc2}(f,k)$ is connected, then either R is an $r(r+1)$-approximation for the C$\underline{k}$SPM, or the output by Procedure 1, when given $f|_{2^R} \in \mathscr{F}_r^R$ and $k \in [n-1]$ as input, is an $O(n^{r-1}/k^{r-1})$-approximation for the C$\underline{k}$SPM.*

Proof. By Lemma 3.6, we know that $|R| \geq k$ and $f(R) \geq \mathrm{OPT}/(r(r+1))$. If R is a k-subset, then it is a feasible solution of the C$\underline{k}$SPM achieving the approximation ratio $r(r+1)$.

It remains to consider the case where $|R| > k$. Suppose that $S := \textsc{Prc1}(f|_{2^R}, k)$. Then by Lemma 3.5 and Lemma 3.6(b), we see that S is a connected $\underline{k}$-subset of R satisfying

$$f(S) \geq c \cdot \frac{k^r}{|R|^{r-1}} \cdot \min_{v \in R} f(v|R \setminus \{v\}) \geq c \cdot \frac{k^r}{n^{r-1}} \cdot \frac{\mathrm{OPT}}{k(r+1)} \geq \frac{c}{r+1} \cdot \left(\frac{k}{n}\right)^{r-1} \cdot \mathrm{OPT},$$

which proves the lemma. □

The preceding lemma establishes an $O(n^{r-1}/k^{r-1})$-approximation for the C$\underline{k}$SPM instances with $f \in \mathscr{F}_r^V$, when R is connected. The next subsection addresses the complementary case in which R is not connected. In this setting, we still have $|R| \geq k$ (by Lemma 3.6), but (by the execution of Procedure 2) no connected subset of R contains k or more elements. R is the disjoint union of its maximal connected subsets, each of which is often referred to as a *connected component* of R.

3.3 Component merging

In this subsection, we assume that the set R returned by Procedure 2 is not connected, and each connected component of R contains no more than $k-1$ elements.

As established in Lemma 3.6, $f(R)$ has been a constant approximation of the optimal objective value OPT:

$$f(R) \geq \frac{\text{OPT}}{r+1}.$$

Due to the function's decomposable structure, $f(R)$ equals the sum of the function values $f(C)$ over the connected components C of R. If R consists of a small number of connected components, then the component with the highest value serves as a good approximation for the C$\underline{k}$SPM problem. The difficulty arises when R is partitioned into a large number of connected components, most of which have a small cardinality.

- A connected component S of R is called *small* if $|S| < k/3$, and *big* otherwise.
- Let $\mathbb{S}$ (resp. $\mathbb{B}$) denote the set of all small (resp. big) connected components of R. We often call each member of $\mathbb{S}$ a *small component* for short.

$$\sum_{S \in \mathbb{S} \cup \mathbb{B}} f(S) = f(R) \geq \frac{\text{OPT}}{r+1}. \tag{5}$$

Our basic idea is to group the small components in $\mathbb{S}$. For each group, we add some additional elements to connect them, forming a single connected subset. On one hand, we need to ensure that the number of groups is not too large, thereby overcoming the aforementioned difficulty. On the other hand, we must guarantee that each of these connected subsets is a $\underline{k}$-subset.

The technical challenge lies in how to perform the grouping. To control the number of groups, a natural idea is to ensure that the total cardinality of the members (components in $\mathbb{S}$) in each group is not too small. However, considering this alone is insufficient. This is because connecting the group members requires adding extra elements, and the increase in cardinality from these additions depends on the graphical structure. For example, components that are "far apart" would require a large number of additional elements to connect. If they were grouped together, their original total cardinality must not be too large (due to the cardinality constraint k). In order to group small components in $\mathbb{S}$ that are "close" to each other as much as possible, we need to utilize the positional information of these small connected components within the graph G. To do this, we shrink each small component in $\mathbb{S}$ into a single node and find a spanning tree in the resulting graph. Using the hierarchical structure and relative positions of these nodes in the spanning tree, we can group and connect (merge) the small connected components that are relatively close to each other, thereby keeping both the number of groups and the cardinality of the connected subsets within appropriate ranges. This enables us to obtain a set $\mathbb{M}$ of $O(n/k)$ connected $\underline{k}$-subsets that collectively cover R (see Procedure 3). By choosing the best connected $\underline{k}$-subset in $\mathbb{M} \cup \mathbb{B}$, we derive an $O(n/k)$-approximation for the C$\underline{k}$SPM in the case of disconnected R (see Theorem 3.8).

Shrinking. For convenience, we often use $|G|$ to denote the number of vertices in the graph $G = (V, E)$. Given any connected subset S of V, *shrinking* S in G means gluing all vertices in S together to form a new vertex v_S such that all vertices of $V \setminus S$ and all edges in E with both ends in $V \setminus S$ are unchanged, each edge joining a vertex $u \in V \setminus S$ and a vertex in S becomes an edge joining u and v_S, and all edges with both ends in S are deleted. We say that v_S *represents* (the vertices of) S.

- Let $G/\mathbb{S}$ be the graph derived from G by shrinking all members of $\mathbb{S}$, where each vertex outside small components in $\mathbb{S}$ *represents* itself.

To distinguish between the vertices in G and those in $G/\mathbb{S}$, we refer to the latter as "nodes." For any subgraph H of $G/\mathbb{S}$, let $\Lambda(H)$ denote the set of vertices in V that are represented by the nodes in H.

BFS spanning tree. Starting from an arbitrary node t in $G/\mathbb{S}$, a breadth first search (BFS) over G constructs a spanning tree T of $G/\mathbb{S}$ *rooted* at t, along with a *reverse BFS ordering* $v_1, v_2, \ldots, v_{|T|}$ of its nodes, such that

- $v_{|T|}$ is the root t, and each node v_i $(1 \leq i \leq |T|)$ has its index i lower than its parent.

For any subtree T' of T that contains $v_{|T|}$, and any node v_i in T', we also considered T' being rooted at $v_{|T|}$, and use T'_i to denote the (maximal) subtree of T' rooted at v_i, consisting of v_i and all its descendants in T'. It is clear that $T' \setminus T'_i$ is a subtree of T that contains $v_{|T|}$ unless $i = |T|$ (in which case, $T' \setminus T'_{|T|} = T' \setminus T' := \emptyset$).

Merging by tree pruning. The for-loop (Steps 6 –15) of Procedure 3 below merges small components to form the set $\mathbb{M}$, via pruning the spanning tree T in a bottom-up fashion (from leaves to root). Each time a subtree is removed or several branches are pruned, the removed part corresponds to a connected $\underline{k}$-subset, which is added to $\mathbb{M}$. The for-loop iterates through the nodes of the tree in order $v_1, v_2, \ldots, v_{|T|}$, considering the subtree rooted at each v_i in turn. For each such subtree, the procedure works as follows:

- If the cardinality of the vertex set represented by the current subtree is less than $k/3$, the procedure does nothing and proceeds to the next node v_{i+1}.
- If the cardinality falls within $[k/3, k]$, the procedure removes the subtree rooted at v_i (Step 13) and places the vertex set it represents into $\mathbb{M}$ (Step 12).
- If the cardinality exceeds k, the procedure identifies a set of subtrees rooted at some children of v_i such that the total number of vertices they represent is between $k/3$ and $2k/3$ (Step 8). The procedure then adds the set composed of these vertices, along with the vertices represented by v_i, to $\mathbb{M}$ (Step 9), and removes these child subtrees (Step 10; note that v_i itself is not removed at this stage). This process of pruning subtrees rooted at some children of v_i and expanding $\mathbb{M}$ is repeated until the number of vertices represented by the subtree rooted at v_i is at most k.

After handling the subtree rooted at v_i in this way, the procedure continues to v_{i+1} and repeats the above steps.

Procedure 3: Small components merging for connected $\underline{k}$-subsets

Input: $\overline{k}$-subset $R = \textsc{Prc2}(f, k)$ that is not connected

1 $\mathbb{S} \leftarrow$ the set of small connected components of R;
2 $\mathbb{B} \leftarrow$ the set of connected components of R that are not small;
3 $\mathbb{M} \leftarrow \emptyset$;
4 **if** $\mathbb{S} \neq \emptyset$ **then**
5 $T \leftarrow$ a spanning tree of $G/\mathbb{S}$ with reverse BFS ordering $v_1, \ldots, v_{|T|}$ of its nodes;
6 **for** $i = 1$ **to** $|T|$ **do**
7 **while** $|\Lambda(T_i)| > k$ **do**
8 $C \leftarrow$ a set of some children of v_i in T_i such that $k/3 \leq \sum_{v_h \in C} |\Lambda(T_h)| \leq 2k/3$;
  `// See the proof of Lemma 3.8 for the construction of` C.
9 $\mathbb{M} \leftarrow \mathbb{M} \cup \{\Lambda(v_i) \cup (\bigcup_{v_h \in C} \Lambda(T_h))\}$;
10 $T \leftarrow T \setminus (\bigcup_{v_h \in C} T_h)$;
11 **if** $k/3 \leq |\Lambda(T_i)| \leq k$ **then**
12 $\mathbb{M} \leftarrow \mathbb{M} \cup \{\Lambda(T_i)\}$;
13 $T \leftarrow T \setminus T_i$;
14 **if** $T \neq \emptyset$ **then**
15 $\mathbb{M} \leftarrow \mathbb{M} \cup \{\Lambda(T)\}$;
16 **Output** $\textsc{Prc3}(R) :=$ any set in $\operatorname{argmax}_{S' \in \mathbb{M} \cup \mathbb{B}} f(S')$

Lemma 3.8. *Given disconnected $R = \textsc{Prc2}(f, k)$, Procedure 3 runs in $O(m)$ time and satisfies the following statements:*

(i) The procedure produces a set $\mathbb{M}$ of connected $\underline{k}$-subsets such that $|\mathbb{M}| \le 3n/k + 1$ and $\sum_{S\in\mathbb{M}} f(S) \ge \sum_{S\in\mathbb{S}} f(S)$.

(ii) The procedure outputs a connected $\underline{k}$-subset $M = \textsc{Prc3}(R)$ such that $(r+1)\cdot(6n/k+1)\cdot f(M) \ge \mathrm{OPT}$.

Proof. To establish the correctness and runtime of Procedure 3, let d_i denote the degree of node v_i in the initial spanning tree computed at Step 5. We observe that each v_i can only be eliminated from the tree T at or after the end of i-th iteration of the for-loop. We prove by induction on i the following claim for the i-th iteration of the for-loop:

(C1) when $|\Lambda(T_i)| > k$, the children set C as specified in Step 8 can be found in $O(d_i)$ time; and

(C2) at the end of the iteration, either v_i is eliminated from T, or $|\Lambda(T_i)| < k/3$.

The base case corresponds to the leaves of the spanning tree and holds true, because, by the definition of $\mathbb{S}$, we have $|\Lambda(v)| < k/3$ for each node $v \in G/\mathbb{S}$.

Suppose the claim (consisting of (C1) and (C2)) holds for all indices smaller than i. If $|\Lambda(T_i)| < k/3$ at the start of the ith iteration, then the iteration has done nothing and the claim is true. If $k/3 \le |\Lambda(T_i)| \le k$ at the start of the i-th iteration, then the while-loop has done nothing and v_i is deleted from the tree at the end of the i-th iteration (Step 13). Now we consider the cases that $|\Lambda(T_i)| > k$ at the start of the ith iteration, for which the while-loop is executed. Let $v_{i_1}, \ldots, v_{i_c}$ be all the children of v_i at that time. By the property of the reverse BFS ordering, the indices $i_1, \ldots, i_c$ of these children are all smaller than i. It follows from the induction hypothesis that

$$|\Lambda(T_{i_j})| < k/3 \text{ for all } j \in [c].$$

Since $|\Lambda(T_i)| > k$ and $|\Lambda(v_i)| < k/3$, we have $\sum_{j=1}^{c} |\Lambda(T_{i_j})| = |\Lambda(T_i)| - |\Lambda(v_i)| > 2k/3$. Let c' be the smallest index such that $\sum_{j=1}^{c'} |\Lambda(T_{i_j})| \ge 2k/3$. Then $|\Lambda(T_{i_{c'}})| < k/3$ enforces $k/3 < \sum_{j=1}^{c'-1} |\Lambda(T_{i_j})| \le 2k/3$. So we can set $C = \{v_{i_1}, \ldots, v_{i_{c'-1}}\}$, which, along with $c < d_i$, justifies (C1). Consider the last execution of the while-loop in the i-th iteration of the for-loop. At the beginning of the execution, Step 7, the set $\Lambda(T_i)$ has cardinality $|\Lambda(T_i)| > k$. After removing a $\overline{2k/3}$-subset $\bigcup_{v_h\in C} \Lambda(T_h)$ at Step 10, it becomes a $\underline{k}$-subset, from which we deduce that $k/3 \le |\Lambda(T_i)| \le k$ holds at the end of the while-loop. In turn, Step 13 eliminates T_i, which includes v_i. This proves (C2) and, consequently, the claim, thereby confirming the correctness of Procedure 3. The runtime follows from the fact that $\sum_{i=1}^{|G/\mathbb{S}|} d_i = 2(|G/\mathbb{S}| - 1) \le 2n$ and it takes $O(m)$ time for Step 5 to construct a BFS spanning tree in $G/\mathbb{S}$.

By the claim, it is straightforward to check that each subset we put into $\mathbb{M}$ is a connected $\underline{k}$-subset. Recall that initially $|\Lambda(T)| = n$. Each time we put a set in $\mathbb{M}$, we must immediately delete a rooted subtree from T (at Step 13) or delete several subtrees rooted at some children of the same node from T (at Step 10). In either case, $|\Lambda(T)|$ decreases by at least $k/3$. It follows that $|\mathbb{M}| \le 3n/k + 1$.

By the construction of $G/\mathbb{S}$, each small connected component in $\mathbb{S}$ is contained in some connected $\underline{k}$-subset in $\mathbb{M}$. Then, since these small connected components are pairwise disjoint, we have

$$\sum_{S\in\mathbb{M}} f(S) \ge \sum_{S\in\mathbb{M}} f\Big(\bigcup_{S'\in\mathbb{S}:S'\subseteq S} S'\Big) \ge \sum_{S\in\mathbb{M}} \sum_{S'\in\mathbb{S}:S'\subseteq S} f(S') \ge \sum_{S'\in\mathbb{S}} f(S'),$$

where the first inequality is implied by f's monotonicity, and the second is by f's supermodularity and normality. Statement (i) is proved.

To see statement (ii), we recall that the members of $\mathbb{B}$ are pairwise disjoint $\overline{k/3}$-subsets. So $|\mathbb{B}| \le n/(k/3) = 3n/k$, and by (i), S is a connected $\underline{k}$-subset of V. Moreover, recalling (5), we have

$$\sum_{S\in\mathbb{M}\cup\mathbb{B}} f(S) \ge \sum_{S\in\mathbb{S}\cup\mathbb{B}} f(S) \ge \frac{\mathrm{OPT}}{r+1}.$$

It implies that

$$\mathrm{OPT} \le (r+1) \sum_{S\in\mathbb{M}\cup\mathbb{B}} f(S) \le (r+1)(|\mathbb{M}| + |\mathbb{B}|) \cdot \max_{S\in\mathbb{M}\cup\mathbb{B}} f(S) \le (r+1)\cdot(6n/k+1)\cdot f(M),$$

establish (ii). □

3.4 Integration

In this subsection, we first integrate Procedures 1 – 3 into an algorithm (Algorithm 4) for solving the C$\underline{k}$SPM problem, achieving an approximation ratio of $O(n^{r-1}/k^{r-1})$. Then, we combine this algorithm with the star-exploration algorithm (Algorithm 3), which has an approximation ratio $O(k^{r-1})$, to provide an $O(n^{(r-1)/2})$-approximation for the C$\underline{k}$SPM problem.

In Algorithm 4 below, we first run Procedure 2, removing removable elements one by one from V until no removable elements remain or only k elements are left. The resulting set R is a $\overline{k}$-subset. If R is a connected k-subset (Step 3), it is returned as the output of the algorithm. If R is connected but contains more than k elements (Step 5), we run Procedure 1. Starting from any connected $\lceil (r-1)k/r \rceil$-subset of R, the "sunflower seed disc", we grow it into a connected $\underline{k}$-subset, "sunflower", through petal attachments, and this subset is returned as the output. If R is not connected (Step 7), we run Procedure 3 to merge small connected components of R, forming a "small" number of connected $\underline{k}$-subsets. Among these subsets and the big connected components of R, the best subset is selected as the algorithm's output.

Algorithm 4: Removing-Attaching-Merging for connected $\underline{k}$-subsets
Input: $f \in \mathscr{F}_r^V$ and $k \in [n-1]$
1 $R \leftarrow$ Prc2(f, k);
2 **if** R *is connected* **then**
3 **if** $|R| = k$ **then**
4 $S \leftarrow R$;
5 **else**
6 $S \leftarrow$ Prc1$(f|_{2^R}, k)$;
7 **else**
8 $S \leftarrow$ Prc3(R);
9 **Output** S

Theorem 3.9. *Algorithm 4 runs in $O(mn^2)$ time and achieves an approximation ratio of $O(n^{r-1}/k^{r-1})$ for the C$\underline{k}$SPM problem on $\mathscr{F}_r$, provided $k > r$.*

Proof. The runtimes of Procedures 1, 2 and 3 are $O(mn^2)$, $O(n^2)$, and $O(m)$, respectively, as specified in Lemmas 3.5, 3.6 and 3.8. Thus, the overall runtime of the algorithm is $O(mn^2)$. The approximation ratio $O(\max\{r(r+1), n^{r-1}/k^{r-1}, (r+1)(6n/k+1)\})$, i.e., $O(n^{r-1}/k^{r-1})$, follows immediately from Lemmas 3.7 and 3.8. □

Similar to the analysis of Algorithm 3, when evaluating the performance of Algorithm 4, we can replace the optimal value OPT for the C$\underline{k}$SPM problem in inequality (4) with the optimal value OPT* for the $\underline{k}$SPM problem, and hence replace every occurrence of OPT in the subsequent analysis with OPT*. This implies that, if V is connected, then Algorithm 4 in fact outputs an $O(n^{r-1}/k^{r-1})$-approximation for $\underline{k}$SPM, which is connected.

Corollary 3.10. *Provided that $k > r$ and V is connected, Algorithm 4 outputs an $O(n^{r-1}/k^{r-1})$-approximate solution for the $\underline{k}$SPM problem on $\mathscr{F}_r$, which is connected.*

The combination of Theorems 3.3 and 3.9 gives the following main result of this section. Taking the better solution output by Algorithms 3 and 4 (when $k > r$), and using brute-force search to find the optimum (in case of $k \le r$), we can guarantee an approximation ratio of $O(\min\{k^{r-1}, n^{r-1}/k^{r-1}\})$ with $\min\{k^{r-1}, n^{r-1}/k^{r-1}\} \le n^{(r-1)/2}$.

Theorem 3.11. *There is an $O(n^{(r-1)/2})$-approximation algorithm for the C$\underline{k}$SPM (resp. CkSPM) problem on $\mathscr{F}_r$, which runs in $O(mn^{r+1})$ time.*

Combined with Corollaries 3.4 and 3.10, we obtain the following corollary.

Corollary 3.12. *Suppose that V is connected. Then there is a polynomial-time algorithm that outputs an $O(n^{(r-1)/2})$-approximate solution for the $\underline{k}$SPM (resp. kSPM) problem on $\mathscr{F}_r$, which is connected.*

Taking $r = 2$ in the above theorem, our algorithm improves the approximation ratio for the densest connected k-subgraph (DCkS) problem from the previously best $O(n^{2/3})$ [16] to $O(n^{1/2})$.

Corollary 3.13. *There is an $O(\sqrt{n})$-approximation algorithm for the DCkS problem, which runs in $O(mn^3)$ time.*

When $r = 2$, the performance ratio $O(\sqrt{n})$ in Corollary 3.12 is best possible, because, as shown for the DCkS problem [16], the ratio between the maximum density of k-subgraph and the maximum density of a connected k-subgraph can be as high as $\Omega(\sqrt{n})$.

4 Concluding remarks

In this paper, we study the cardinality-constrained supermodular maximization problems, addressing versions both with and without a connectivity requirement. We present polynomial-time approximation algorithms for both sets of problems, assuming the underlying nonnegative supermodular function is monotone and r-decomposable, with r being a constant. Our algorithms yield two complementary approximation ratios: $O(k^{r-1})$ and $O(n^{r-1}/k^{r-1})$, which combine to achieve an overall $O(n^{(r-1)/2})$-approximation ratio.

4.1 Approximation for $\underline{k}$SPM on non-monotone functions

Although for monotone functions, we can solve the $\underline{k}$SPM problem using the algorithm for the kSPM problem, this approach generally does not work for non-monotone functions. However, a slight modification to the greedy peeling algorithm, Algorithm 1 (originally designed for kSPM), enables us to handle the $\underline{k}$SPM problem for non-monotone functions. Let Algorithm 1' denote the modification of Algorithm 1, where Step 5 is replaced with "**Output** S_k or $\emptyset$ whichever has the larger function value".

Corollary 4.1. *Algorithm 1' achieves an approximation ratio of $O(n^{r-1}/k^{r-1})$ for the $\underline{k}$SPM problem on nonnegative supermodular functions that are r-decomposable, provided $r < k$.*

Indeed, in the proof of Theorem 2.3, the monotonicity of the function f is used only in the first case, where $f(v_i|S_i \setminus \{v_i\}) \leq \mathrm{OPT}/(2k)$ for each $i \in [k+1, n]$. Because f is nonnegative and supermodular,

$$f(A) \leq f(A) + f(B \setminus A) \leq f(B) + f(\emptyset) \leq 2\max\{f(B), f(\emptyset)\} \text{ holds for all } A \subseteq B \subseteq V.$$

Thus, for the case, we now have

$$\frac{\mathrm{OPT}}{\max\{f(S_k), f(\emptyset)\}} \leq \frac{2 \cdot \mathrm{OPT}}{f(S_0')} \leq 4,$$

as a substitute for the conclusion $\mathrm{OPT}/f(S_k) \leq 2$ drawn for the case in monotone setting. The remaining arguments in the proof of Theorem 2.3 remain valid for non-monotone functions, and the $O(n^{r-1}/k^{r-1})$ approximation ratio in Corollary 4.1 follows.

4.2 Optimization under tight connectivity

Beyond the standard approach to defining connectedness of subsets (which we have discussed so far), there is another natural, yet stronger, notion of tight connectivity. Given r-decomposition $(V_i, f_i)_{i=1}^m$ of function $f \in \mathscr{F}_r$, we call a subset S of V *tightly connected* if the hypergraph with vertex set S and hyperedge set $\{V_i : S \subseteq V_i, i \in [m]\}$ is connected. Clearly, if S is tightly connected, then S is connected; however, the converse is not necessarily true. Moreover, for the ground set V, connectedness and tight connectedness coincide.

We prove in Appendix C that the supermodular maximization for finding a maximum-valued tightly connected $\underline{k}$-subset preserves the asymptotic approximation ratio achievable for the $\underline{k}$SPM problem. In particular, the connectivity property stated in Corollary 3.12 is strengthened to tight connectivity, as highlighted by the following corollary.

Corollary 4.2. *Suppose that V is connected. Then there is a polynomial-time algorithm that outputs an $O(n^{(r-1)/2})$-approximate solution for the $\underline{k}$SPM problem on $\mathscr{F}_r$, which is tightly connected.*

4.3 Future research directions

This work has focused on maximizing over supermodular functions with cardinality and connectivity constraints. Richer constraints open several intriguing research directions. Two of the most natural extensions are knapsack and matroid constraints.

Knapsack constraints. In many practical applications, such as resource allocation and budgeting problems, a simple cardinality constraint may be too restrictive. A more generalized form of the problem is to consider a knapsack constraint: assign a nonnegative weight $w(v)$ to each element $v \in V$, and require that the weighted sum of the subset $S \subseteq V$, $\sum_{v \in S} w(v)$, does not exceed a given budget W. Note that the cardinality constraint is a special case where $W = k$ and $w(v) = 1$ for all $v \in V$. Sviridenko [50] has studied the problem of submodular maximization under knapsack constraints.

Matroid constraints. Matroids provide a rich generalization of independence structures and have been widely used in combinatorial optimization. Matroid constraints often arise in network design and machine learning applications, where dependencies between elements need to be captured. Given a matroid $(V, \mathcal{I})$ consisting of a finite set V and a non-empty collection $\mathcal{I}$ of subsets of V. The matroid constraint restricts feasible solutions to members of $\mathcal{I}$. Note that the cardinality constraint corresponds to the uniform matroid, i.e., $\mathcal{I} = \{S \subseteq V : |S| \leq k\}$. Calinescu et al. [12] has studied the problem of submodular maximization under matroid constraints.

Exploring supermodular maximization under these richer constraint classes promises both theoretical challenges and practical impact.

References

[1] N. Alon, S. Arora, R. Manokaran, D. Moshkovitz, O. Weinstein, Inapproximability of densest k-subgraph from average case hardness, Manuscript (2011). URL: `http://www.cs.princeton.edu/~rajsekar/papers/dks.pdf`.

[2] E. Althaus, M. Blumenstock, A. Disterhoft, A. Hildebrandt, M. Krupp, Algorithms for the maximum weight connected k-induced subgraph problem, in: Combinatorial Optimization and Applications (COCOA), 2014, pp. 268–282.

[3] R. Andersen, K. Chellapilla, Finding dense subgraphs with size bounds, in: Algorithms and models for the web-graph, volume 5427 of *Lecture Notes in Comput. Sci.*, 2009, pp. 25–37.

[4] B. Applebaum, B. Barak, A. Wigderson, Public-key cryptography from different assumptions, in: Proceedings of the 2010 ACM International Symposium on Theory of Computing (STOC), 2010, pp. 171–180.

[5] S. Arora, D. Karger, M. Karpinski, Polynomial time approximation schemes for dense instances of NP-hard problems, J. Comput. Syst. Sci. 58 (1999) 193–210.

[6] Y. Asahiro, K. Iwama, H. Tamaki, T. Tokuyama, Greedily finding a dense subgraph, J. Algorithms 34 (2000) 203–221.

[7] W. Bai, J. Bilmes, Greed is still good: Maximizing monotone Submodular+Supermodular (BP) functions, in: Proceedings of the 35th International Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, 2018, pp. 304–313.

[8] A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, A. Vijayaraghavan, Detecting high log-densities—an $O(n^{1/4})$ approximation for densest k-subgraph, in: Proceedings of the 2010 ACM International Symposium on Theory of Computing (STOC), 2010, pp. 201–210.

[9] A. Bhaskara, M. Charikar, V. Guruswami, A. Vijayaraghavan, Y. 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 (SODA), 2012, pp. 388–405.

[10] M. Braverman, Y. K. Ko, A. Rubinstein, O. Weinstein, ETH hardness for densest-k-subgraph with perfect completeness, in: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2017, pp. 1326–1341.

[11] N. Buchbinder, M. Feldman, J. Naor, R. Schwartz, A tight linear time (1/2)-approximation for unconstrained submodular maximization, SIAM J. Comput. 44 (2015) 1384–1402.

[12] G. Calinescu, C. Chekuri, M. Pál, J. Vondrák, Maximizing a monotone submodular function subject to a matroid constraint, SIAM J. Comput. 40 (2011) 1740–1766.

[13] L. F. O. Chamon, A. Amice, A. Ribeiro, Approximately supermodular scheduling subject to matroid constraints, IEEE Trans. Automat. Control 67 (2022) 1384–1396.

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

[15] D. Z. Chen, R. Fleischer, J. Li, Densest k-subgraph approximation on intersection graphs, in: Approximation and online algorithms, volume 6534 of *Lecture Notes in Comput. Sci.*, 2011, pp. 83–93.

[16] X. Chen, X. Hu, C. Wang, Finding connected k-subgraphs with high density, Inf. Comput. 256 (2017) 160–173.

[17] E. Chlamtáč, M. Dinitz, C. Konrad, G. Kortsarz, G. Rabanca, The densest k-subhypergraph problem, SIAM J. Discrete Math. 32 (2018) 1458–1477.

[18] D. G. Corneil, Y. Perl, Clustering and domination in perfect graphs, Discrete Appl. Math. 9 (1984) 27–39.

[19] E. D. Demaine, M. Hajiaghayi, K.-i. Kawarabayashi, Algorithmic graph minor theory: decomposition, approximation, and coloring, in: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2005, pp. 637–646.

[20] D. S. dos Santos, K. Klamroth, P. Martins, L. Paquete, Solving the multiobjective quasi-clique problem, Eur. J. Oper. Res. 323 (2025) 409–424.

[21] U. Feige, Relations between average case complexity and approximation complexity, in: Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing (STOC), 2002, pp. 534–543.

[22] U. Feige, G. Kortsarz, D. Peleg, The dense k-subgraph problem, Algorithmica 29 (2001) 410–421.

[23] U. Feige, M. Langberg, Approximation algorithms for maximization problems arising in graph partitioning, J. Algorithms 41 (2001) 174–211.

[24] U. Feige, V. S. Mirrokni, J. Vondrák, Maximizing non-monotone submodular functions, SIAM J. Comput. 40 (2011) 1133–1153.

[25] M. Grötschel, L. Lovász, A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, Combinatorica 1 (1981) 169–197.

[26] Q. Han, Y. Ye, J. Zhang, An improved rounding method and semidefinite programming relaxation for graph partition, Math. Program. 92 (2002) 509–535.

[27] R. Hassin, S. Rubinstein, A. Tamir, Approximation algorithms for maximum dispersion, Oper. Res. Lett. 21 (1997) 133–137.

[28] S. Hu, X. Wu, T.-H. H. Chan, Maintaining densest subsets efficiently in evolving hypergraphs, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, p. 929–938.

[29] D. A. Iancu, M. Sharma, M. Sviridenko, Supermodularity and affine policies in dynamic robust optimization, Oper. Res. 61 (2013) 941–956.

[30] C. Jones, A. Potechin, G. Rajendran, J. Xu, Sum-of-squares lower bounds for densest k-subgraph, in: Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC), 2023, pp. 84–95.

[31] S. Khot, Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique, SIAM J. Comput. 36 (2006) 1025–1071.

[32] S. Khuller, B. Saha, On finding dense subgraphs, in: Automata, languages and programming. Part I, volume 5555 of *Lecture Notes in Comput. Sci.*, 2009, pp. 597–608.

[33] C. Komusiewicz, M. Sorge, K. Stahl, Finding connected subgraphs of fixed minimum density: Implementation and experiments, in: International Symposium on Experimental Algorithms (SEA), 2015, pp. 82–93.

[34] G. Kortsarz, D. Peleg, On choosing a dense subgraph, in: Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science (FOCS), 1993, pp. 692–701.

[35] M. Liazi, I. Milis, F. Pascual, V. Zissimopoulos, The densest k-subgraph problem on clique graphs, J. Comb. Optim. 14 (2007) 465–474.

[36] M. Liazi, I. Milis, V. Zissimopoulos, Polynomial variants of the densest/heaviest k-subgraph problem, in: Proceedings of the 20th British Combinatorial Conference, Durham, 2005.

[37] M. Liazi, I. Milis, V. Zissimopoulos, A constant approximation algorithm for the densest k-subgraph problem on chordal graphs, Inform. Process. Lett. 108 (2008) 29–32.

[38] L. Lovász, Submodular functions and convexity, in: Mathematical programming: the state of the art, 1983, pp. 235–257.

[39] P. Manurangsi, Almost-polynomial ratio ETH-hardness of approximating densest k-subgraph, in: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC), 2017, pp. 954–961.

[40] M. Narasimhan, J. Bilmes, A submodular-supermodular procedure with applications to discriminative structure learning, in: Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence, 2005, pp. 404–412.

[41] G. L. Nemhauser, L. A. Wolsey, Best algorithms for approximating the maximum of a submodular set function, Math. Oper. Res. 3 (1978) 177–188.

[42] G. L. Nemhauser, L. A. Wolsey, M. L. Fisher, An analysis of approximations for maximizing submodular set functions. I, Math. Program. 14 (1978) 265–294.

[43] T. Nonner, PTAS for densest k-subgraph in interval graphs, Algorithmica 74 (2016) 528–539.

[44] D. Paccagnan, J. R. Marden, Utility design for distributed resource allocation—Part II: Applications to submodular, covering, and supermodular problems, IEEE Trans. Automat. Control 67 (2022) 618–632.

[45] Y. Perl, Y. Shiloach, Efficient optimization of monotonic functions on trees, SIAM J. Algebraic Discrete Methods 4 (1983) 512–516.

[46] P. Raghavendra, D. Steurer, Graph expansion and the unique games conjecture, in: Proceedings of the 2010 ACM International Symposium on Theory of Computing (STOC), 2010, pp. 755–764.

[47] S. S. Ravi, D. J. Rosenkrantz, G. K. Tayi, Heuristic and special case algorithms for dispersion problems, Oper. Res. 42 (1994) 299–310.

[48] A. Schrijver, A combinatorial algorithm minimizing submodular functions in strongly polynomial time, J. Combin. Theory Ser. B 80 (2000) 346–355.

[49] A. Srivastav, K. Wolf, Finding dense subgraphs with semidefinite programming, in: International Workshop on Approximation Algorithms for Combinatorial Optimization, 1998, pp. 181–191.

[50] M. Sviridenko, A note on maximizing a submodular set function subject to a knapsack constraint, Oper. Res. Lett. 32 (2004) 41–43.

[51] Z. Svitkina, L. Fleischer, Submodular approximation: sampling-based algorithms and lower bounds, SIAM J. Comput. 40 (2011) 1715–1737.

[52] D. M. Topkis, Supermodularity and Complementarity, Princeton University Press, Princeton, NJ, 1998.

A The arbitrarily large approximation ratio of the single-element augmentation for 2-decomposable functions

The algorithm proposed by Bai and Bilmes [7] starts from the empty set and greedily adds a single element with the highest marginal value at each step (see Algorithm 5).

Algorithm 5: Greedy algorithm for kSPM by Bai and Bilmes [7]

Input: A supermodular function $f : 2^V \to \mathbb{R}_+$ and an integer $k \in [n]$

$S_0 \leftarrow \emptyset$;

for $i = 1$ **to** k **do**

  $s_i \leftarrow$ any element $\in \operatorname{argmax}_{v \in V \setminus S_{i-1}} \{f(v|S_{i-1})\}$;

  $S_i \leftarrow S_{i-1} \cup \{s_i\}$;

Output S_k

The approximation ratio of this algorithm for monotone 2-decomposable nonnegative supermodular functions can be arbitrarily large, even for a ground set of fixed size $n \geq 4$. To see this, consider the function $f : 2^V \to \mathbb{R}_+$ defined on $V = \{v_1, v_2, \ldots, v_n\}$ as follows

$$f(S) = M \cdot n_1(S) \cdot (n_1(S) - 1) + n_2(S) \text{ for all } S \subseteq V,$$

where $M \gg 1$, $n_1(S) = |S \cap \{v_1, v_2\}|$, $n_2(S) = |S \cap \{v_3, v_4, \ldots, v_n\}|$. It is routine to check that $f \in \mathscr{F}_2$, where its 2-decomposition is $(2; (\{v_1, v_2\}, f_1), (\{v_{i+1}\}, f_i)_{i=2}^{n-1})$ with $f_1(S) = M|S|(|S| - 1)$ for all $S \subseteq \{v_1, v_2\}$ and $f_i(S) = |S|$ for all $i \in \{2, \ldots, n-1\}$ and $S \subseteq \{v_{i+1}\}$.

For each $2 \leq k \leq n - 2$, the optimal solution of the kSPM is $S^* = \{v_1, v_2, \ldots, v_k\}$ with $f(S^*) = 2M + k - 2$. However, since $f(v_1|S) = f(v_2|S) = 0$ for any $S \subseteq V$ with $S \cap \{v_1, v_2\} = \emptyset$, the algorithm never adds v_1 or v_2 to S_i during its execution. Consequently, the solution S_k obtained by this algorithm satisfies $S_k \cap \{v_1, v_2\} = \emptyset$, resulting in $f(S_k) = k$ and an arbitrarily large approximation ratio $\Theta(M/k)$.

B Normalization assumption

The following proposition shows that normalizing a nonnegative set function f does not decrease the approximation ratio for the maximization problem $\max_{S \in \mathcal{F}} f(S)$, and preserves the function's property of being nonnegative, monotone, supermodular (and r-decomposable).

Proposition B.1. *Given nonnegative function $f : 2^V \to \mathbb{R}_+$, let $g : 2^V \leftarrow \mathbb{R}$ be the normalized function defined by $g(S) = f(S) - f(\emptyset)$ for all $S \subseteq V$. Then the following hold.*

(i) $f(T)/f(S) \leq g(T)/g(S)$ for all $S, T \subseteq V$ with $f(S) \leq f(T)$.

(ii) g is nonnegative and monotone if f is monotone.

(iii) g is supermodular if f is supermodular.

(iv) g is r-decomposable if f is monotone, supermodular and r-decomposable.

Proof. It suffices to consider the case of $f(\emptyset) > 0$. The inequality in (i) follows from $g(T)f(S) - f(T)g(S) = (f(T) - f(\emptyset))f(S) - f(T)(f(S) - f(\emptyset)) = f(\emptyset) \cdot (f(T) - f(S)) \geq 0$. Statements (ii) and (iii) are trivial.

To prove (iv), we assume that $V = [n]$ and f admits r-decomposition $(V_i, f_i)_{i=1}^m$. Let $I_j = \{i \in [m] : j \in V_i\}$. Consider any $j \in V$. Since f is monotone, we have $f(j|\emptyset) = \sum_{i \in I_j} f_i(j|\emptyset) \geq 0$. Therefore, for each $i \in I_j$, there exists $w_{ij} \geq 0$ such that

$$f(j|\emptyset) = \sum_{i \in I_j} w_{ij}.$$

To be specific, when $i \in I_j$, we can set w_{ij} to $f(j|\emptyset)$ if $i = \min I_j$, and to 0 otherwise. Let $g_i : 2^{V_i} \to \mathbb{R}_+$ be the function defined as follows:

$$g_i(S) = f_i(S) - f_i(\emptyset) + \sum_{j \in S} (w_{ij} - f_i(j|\emptyset)) \text{ for each } S \subseteq V_i.$$

We claim that $(V_i, g_i)_{i=1}^m$ is an r-decomposition of g. First, since f_i is supermodular,

$$g_i(S) \geq \sum_{j \in S} f_i(j|\emptyset) + \sum_{j \in S} (w_{ij} - f_i(j|\emptyset)) = \sum_{j \in S} w_{ij} \geq 0 \text{ for each } S \subseteq V_i,$$

saying that g_i is nonnegative. Second, it is easy to see that

$$g_i(A\cup B)+g_i(A\cap B)-g_i(A)-g_i(B) = f_i(A\cup B)+f_i(A\cap B)-f_i(A)-f_i(B) \geq 0 \text{ for all } A, B \subseteq V_i,$$

showing that g_i is supermodular. Finally, f's decomposability implies that for each $S \subseteq V_i$

$$\begin{aligned}\sum_{i=1}^{m} g_i(S \cap V_i) &= \sum_{i=1}^{m} \left(f_i(S \cap V_i) - f_i(\emptyset) + \sum_{j \in S \cap V_i} (w_{ij} - f_i(j|\emptyset)) \right) \\ &= f(S) - f(\emptyset) + \sum_{j \in S} \sum_{i \in I_j} (w_{ij} - f_i(j|\emptyset)) \\ &= g(S)\end{aligned}$$

which justifies (iv). □

C Tight connectivity requirements

Given function $f \in \mathscr{F}_r$ with its r-decomposition $(V_i, f_i)_{i=1}^m$, a subset S of V is tightly connected if and only if the graph with vertex set S and edges set $\cup_{i\in[m]:V_i\subseteq S}\{uv : \{u, v\} \subseteq V_i, u \neq v\}$ is connected.

The tightly connected $\underline{k}$-subset supermodular maximization problem is to find a tightly connected $\underline{k}$-subset S of V with maximum $f(S)$. If $k < r^2 + r$, then we can use a brute-force search to find the optimal tightly connected $\underline{k}$-subset. For $k \geq r^2+r$, the following theorem implies that the supermodular maximization on $\mathscr{F}_r$ subject to cardinality and tight connectivity constraints admits the same approximation ratio (up to a constant) as one derives for the $\underline{k}$SPM problem.

Theorem C.1. *Suppose that $k \geq r^2 + r$. Let S_k^* be the optimal solution of the $\underline{k}$SPM problem on $f \in \mathscr{F}_r$. If there is a polynomial algorithm that finds a connected $\underline{k}$-subset S_k with $f(S_k) \geq \alpha \cdot f(S_k^*)$, then there is a constant c and a polynomial algorithm that finds a tightly connected $\underline{k}$-subset S_k' satisfying $f(S_k') \geq c\alpha \cdot f(S_k^*)$.*

Proof. Let $k' = \lfloor k/r \rfloor$ and $S_{k'}^*$ be any $\underline{k'}$-subset in $\operatorname{argmax}\{f(S) : |S| \leq k', S \subseteq S_k\}$. By Corollary 3.10, we can find a connected $\underline{k'}$-subset $S_{k'} \subseteq S_k$ in polynomial time such that $f(S_{k'}) \geq c_1 \cdot f(S_{k'}^*)$, where c_1 is a constant. By Lemma 2.2, there is a constant c_2 such that $f(S_{k'}^*) \geq c_2 \cdot f(S_k)$.

Recall that $G = (V, E)$ is a graph with edge set $E = \cup_{i=1}^m\{uv : \{u, v\} \subseteq V_i, u \neq v\}$. Since $S_{k'}$ is connected, we have a spanning tree T of the graph $G[S_{k'}]$. By definition, we can construct a function $g : E(T) \to [m]$ satisfying $e \subseteq V_{g(e)}$ for each $e \in E(T)$. Define subset $S_k' := S_{k'} \cup (\bigcup_{e\in E(T)} V_{g(e)})$. Then $|S_k'| \leq |S_{k'}| + (|S_{k'}| - 1)(r - 2) \leq r|S_{k'}| \leq k$ and S_k' is tightly connected. Besides, the monotonicity of f implies

$$f(S_k') \geq f(S_{k'}) \geq c_1 \cdot f(S_{k'}^*) \geq c_1c_2 \cdot f(S_k) \geq c_1c_2\alpha \cdot f(S_k^*),$$

verifying the theorem. □