Supermodularity, Iterative Peeling, and Densest Subgraphs

COMP 5112 Final Project Report

Rebecca Kempe

1 Introduction

1.1 Motivation

Many real-world problems can be formulated as finding clusters in graphs or optimizing density measures on graphs. These problems come from a range of disciplines, spanning linguistics, finance, neuroscience, genetics, and more [18]. For example, one version of community detection [13] is the problem of extracting highly correlated groups of pages from the network structure of a hyperlinked environment. Kleinberg's work [13] aims to improve search by identifying authoritative pages within a group of related web pages. Another problem is large near-clique extraction, where the goal is to cluster content by topic and order the content clusters by their popularity [18]. Similarly, it is possible to analyze the content of a text based on creating "text networks" where nodes in the graph are words and they are connected by an edge if they appear within the same sentence [8, 9]. The groups of words that appear most frequently can be extracted and used to determine the topic of the text. This is formally known in linguistics as "centering resonance analysis" [9].

These problems, as well as larger classes of similar problems, have motivated the study of graph mining techniques such as correlation mining, graph clustering, graph compression, and dense subgraph discovery [14, 18]. In this report, we are most interested in the area of dense subgraph discovery. The canonical problem is the densest subgraph problem (DSP). The DSP can be stated as follows: given a graph $G = (V, E)$, we find a subset $S \subseteq V$ that maximizes $\frac{|E(S)|}{|S|}$, where $E(S)$ is the set of edges with both endpoints in S. Alternatively, we find S that maximizes the average degree in the induced graph $G[S]$. In the broadest sense, the objective is to find a subgraph that maximizes some measure of density. Dense subgraphs can reveal useful structure and information about a graph, and various related problems or "density" measures are studied and applied in different areas. The DSP also has many interesting connections to algorithms and combinatorial optimization. For a recent survey of the DSP and its variants, see [14].

1.2 Prior Works

It is well known that the DSP is polynomial time solvable, and there have been a series of flow-based exact algorithms for finding the densest subgraph [10, 11, 12, 17]. However, these algorithms have proved computationally prohibitive in practice, which has motivated the study of approximation algorithms [4, 7]. Charikar [5] designed an LP-based exact algorithm for solving the DSP, used the dual of this LP to show that the greedy peeling algorithm proposed by Asahiro et al.[1] [1] is a $\frac{1}{2}$-approximation for the DSP, and showed that this algorithm runs in linear time.

Boob et al. extended Charikar's greedy algorithm to an iterative peeling algorithm they called Greedy++ [4], which they conjectured is a $(1-\epsilon)$-approximation to the DSP. They also conjectured that the number of iterations required for the algorithm to reach this approximation would be $T = O\left(\frac{1}{\epsilon^2}\right)$, where T is the number of iterations. However, they were only able to prove the same bound as Charikar. Chekuri et al. [7] were able to prove that Greedy++ is a $(1-\epsilon)$-approximation to the DSP using the lens of supermodularity

[1] This is commonly known as "Charikar's greedy algorithm" or "Charikar's peeling algorithm".

and a generalized algorithm called SuperGreedy++, but with a much weaker bound on runtime. Proving a tight bound on the runtime of Greedy++ is an open question.

1.3 Summary

In this report, we primarily study the DSP through the lens of supermodularity, and we closely follow the work of Chekuri et al. [7]. We state the densest supermodular subset problem (DSSP) and show that the DSP is a special case of the DSSP. We generalize the $\frac{1}{2}$-approximation bound on greedy peeling, basing the analysis on a factor c_f that depends on the supermodular function f, and discuss the implications of this bound for more complex supermodular functions. We also outline a proof of the convergence of iterative peeling for the DSSP, which implies convergence for the DSP as a special case.

2 Preliminaries

Here we review the classes of functions relavant in this report and provide some important results. This section borrows substantially from the lecture and tutorial notes of Bilmes [2, 3]; other references include the book chapter by McCormick [16], course notes by Vondrák [19], and the classic paper by Lovász [15].

2.1 Set Functions

A set function $f : 2^V \to \mathbb{R}$ assigns real values to subsets of a ground set V. We say that f is normalized if $f(\emptyset) = 0$. We say that f is monotone nondecreasing if $A \subseteq B \implies f(A) \leq f(B)$, and monotone increasing if $A \subseteq B \implies f(A) < f(B)$. Reversing the inequalities, we get monotone nonincreasing and decreasing functions, respectively.

We say that f is additive if the valuation of the union of any two disjoint sets in V is equal to the sum of their valuations; that is, f is additive if $f(A \dot\cup B) = f(A) + f(B)$. Subadditivity and superadditivity are defined analogously: f is subadditive if $f(A \dot\cup B) \leq f(A)+f(B)$ and superadditive if $f(A \dot\cup B) \geq f(A)+f(B)$.

Let V be a ground set, and let $f : 2^V \to \mathbb{R}$. It is often useful to expressed the marginal value under f of an element to a set, which is the gain or loss incurred by adding that element to the set. Formally, this is $f(S \cup \{v\}) - f(S)$, where $S \subsetneq V, v \neq S$. There are many common ways to notate this; we follow the lead of [7] and denote the marginal value of v to S by $f(v|S) := f(S \cup \{v\}) - f(S)$, where $S \subsetneq V, v \neq S$.

Subsets $S \subseteq V$ can be identified with vectors in $\{0, 1\}^V$. The characteristic vector of a subset S is given by $\mathbf{1}_S \in \{0, 1\}^V$, where for all $v \in V$, we have that $\mathbf{1}_S(v) = 1$ if $v \in S$ and $\mathbf{1}_S(v) = 0$ otherwise. Thus, we can view a set function f as being a function $f : \{0, 1\}^V \to \mathbb{R}$. Any such function can be extended to a variety of continuous functions $\tilde f : [0, 1]^V \to \mathbb{R}$, where vectors with fractional components are also assigned valuations. A function $\tilde f : [0, 1]^V \to \mathbb{R}$ is called a continuous extension of f if $\tilde f$ is defined everywhere on $[0, 1]^V$ and $\tilde f(x) = f(x)$ for all $x \in \{0, 1\}^V$.

One very important continuous extension is the Lovász extension $\hat f : [0, 1]^V \to \mathbb{R}$ of a set function f. There are many equivalent definitions of $\hat f$. The original paper [15] defines it as follows: let $\mathbf{x}$ be a vector in $[0, 1]^V$. Then

$$\hat f(\mathbf{x}) = \sum_{i=0}^{n} \lambda_i f(S_i),$$

where $\emptyset = S_0 \subsetneq S_1 \subsetneq S_2 \subsetneq \cdots \subsetneq S_n$ is a chain of sets such that $\sum \lambda_i \mathbf{1}_{S_i} = \mathbf{x}$ and $\sum \lambda_i = 1, \lambda_i \geq 0$.

For completeness, we also provide the following definition used in [7]. Given $\mathbf{x} \in [0, 1]^V$ and $\tau \in [0, 1]$, let $S_\tau = \{v \in V : \mathbf{x}_v \geq \tau\}$. Then $\hat f(\mathbf{x}) = \mathbb{E}_\tau[f(S_\tau)]$, where the randomness is over $\tau \in [0, 1]$ drawn uniformly at random.

2.2 Submodular, Supermodular, and Modular Functions

A submodular function is a real-valued set function characterized by diminishing returns. That is, $f : 2^V \to \mathbb{R}$ is submodular if

$$f(v|A) \geq f(v|B) \text{ for all } A \subsetneq B \subsetneq V, \text{ where } v \notin B. \tag{1}$$

Another definition is that a set function $f : 2^V \to \mathbb{R}$ is submodular if

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B), \text{ for all } A, B \subseteq V. \tag{2}$$

It is not immediately obvious that these two definitions are equivalent. We provide a proof for completeness, adapted from [16].

Lemma 2.1. *A set function $f : 2^V \to \mathbb{R}$ satisfies (1) if and only if it satisfies (2).*

Proof. We first show that (2) implies (1). Take $A = S \cup \{v\}$, $B = T$, where $S \subsetneq T \subsetneq V$ and $v \notin T$. Then we obtain $f((S \cup \{v\}) \cup T) + f((S \cup \{v\}) \cap T) \leq f(S \cup \{v\}) + f(T)$. Since $S \subset T$ and $v \notin T$, we get $f(T \cup \{v\}) + f(S) \leq f(S \cup \{v\}) + f(T)$ and rearranging, we obtain $f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$ which implies (1), by definition of marginal value. To show that (1) implies (2), we first rewrite (1) as $f(S \cup \{v\}) - f(T \cup \{v\}) \geq f(S) - f(T)$, where $S \subsetneq T \subsetneq V$ and $v \notin T$, and rewrite (2) as $f(A \cap B) - f(B) \leq f(B) - f(A \cup B)$. We arbitrarily enumerate the elements of $B \setminus A$ as $e_1, e_2, \ldots, e_k$ and note that for $i < k$, $[(A \cap B) \cup \{e_1, e_2, \ldots, e_i\}] \subseteq [A \cup \{e_1, e_2, \ldots, e_i\}] \subsetneq [A \cup \{e_1, e_2, \ldots, e_i\}] \cup \{e_{i+1}\}$. Therefore, the rearranged (1) implies that $f(A \cap B) - f(A) \leq f((A \cap B) \cup \{e_1\}) - f(A \cup \{e_1\})$. Inductively adding one element from $B \setminus A$ and continuing to apply (1), we eventually obtain the rearranged (2), as required. □

Given any submodular function f, we can form a normalized variant f', where $f'(A) = f(A) - f(\emptyset)$. This operation does not affect submodularity nor any maxima or minima of the function f. If f is normalized, this is a generalized notion of subadditivity; if A and B are disjoint, $f(A \cap B) = 0$ and thus we get subadditivity. Thus in some sense we can say that submodularity is a generalization of subadditivity. For the remainder of this report, we will assume that all submodular functions are so normalized.

A set function f is supermodular if and only if $-f$ is submodular. Supermodular functions are characterized by increasing returns and in some sense are a generalization of superadditive functions. We also can obtain definitions of supermodularity by flipping the inequalities in (1) and (2), and the previous statements in this section can be rephrased for supermodular functions. A function is called modular if it is both submodular and supermodular; in some sense, modular functions can be seen as a discrete analogue of linear functions. We also have the following lemma that relates modular, submodular, and supermodular functions:

Lemma 2.2. *If f is modular and g is submodular, then $f - g$ is supermodular.*

Proof. g is submodular, so $g(v|A) \geq g(v|B)$ $(*)$. Since f is modular, $f(v|A) = f(v|B)$ and thus we can subtract $f(v|A)$ from the LHS of $(*)$ and $f(v|B)$ from the RHS of $(*)$ without imbalancing the inequality. Then $g(v|A) - f(v|A) \geq g(v|B) - f(v|B)$. Multiplying both sides by -1 and rearranging, we obtain $f(v|A) - g(v|A) \leq f(v|B) - g(v|B)$ which implies $(f - g)(v|A) \leq (f - g)(v|B)$, as required. □

Recall the Lovász extension $\hat{f} : [0, 1]^V \to \mathbb{R}$ of f. We have the following important theorem from [15]:

Theorem 2.1. *$\hat{f}$ is convex if and only if f is submodular.*

This implies that $\hat{f}$ is concave if and only if f is supermodular.

3 The DSSP and Related Approximation Algorithms [2]

3.1 The DSP is a special case of the DSSP

The densest supermodular subset problem (DSSP) is due to [7] and is stated as follows: given a ground set V and a normalized, non-negative supermodular function $f : 2^V \to \mathbb{R}^{\geq 0}$, we want to find a subset $S \subseteq V$

[2] In this section, we will slightly abuse notation and take $S - v$ to mean $S \setminus \{v\}$; similarly, we will take $S + v$ to mean $S \cup \{v\}$.

that maximizes $\frac{f(S)}{|S|}$. Alternatively, we are trying to find a subset S^* with the optimal density, λ^*, where $\lambda^* = \max_{S \subseteq V} \frac{f(S)}{|S|}$. Letting $f(S) = |E(S)|$, we obtain the DSP. We now show that the DSP is a special case of the DSSP.

Lemma 3.1. *$|E(S)|$ is supermodular.*

Proof. We provide two proofs: first, a standard proof, then a sketch of a more direct approach. The more standard way to see this is that $|E(S)| = \sum_{v \in S} \deg_v(S) - |\delta(S)|$, where $\delta(S)$ is the cut function (edges with exactly one endpoint in S). $\sum_{v \in S} \deg_v(S)$ is clearly modular, since the degree of a vertex is independent of the subgraph it's contained in, and $|\delta(S)|$ is submodular, since increasing the number of vertices decreases the chances that edges with one endpoint at the new vertex v will not have another endpoint in S. Thus $|E(S)|$ is the difference of a modular and a submodular function, and is supermodular by Lemma 2.2. More directly, let $f(S) = |E(S)|$. Then $f(v|S)$ is the number of vertices in S that share an edge with v. If $S = \emptyset$, there are no vertices in S to share an edge with v. As the size of S grows, the number of vertices that share an edge with v either grows or remains the same. Thus $f(S) = |E(S)|$ is supermodular by definition. □

3.2 Peeling for DSP and Extension to DSSP

Charikar's greedy algorithm for solving the DSP can be extrapolated to a greedy peeling algorithm for the DSSP [7]. In this section, we state the algorithm and investigate the associated guarantees.

From [5] we have the following greedy $\frac{1}{2}$-approximation for the DSP:

1. Given a graph, repeatedly remove the vertex with the lowest current degree, as well as all edges attached to it. (Ties are broken arbitrarily.)
2. From this, we get an ordering $v_1, v_2, \ldots, v_n$ of vertices, where v_i is the ith vertex in the removal order.
3. We choose the suffix $S_i = \{v_i, v_{i+1}, \ldots, v_n\}$ that induces the subgraph with the highest density λ.

We would like to convert this into an algorithm for the DSSP. Note that the current degree of any vertex v is the marginal value of v to $S - v$, where S is the current set of remaining vertices. Thus, we can replace the step of removing the vertex with the lowest current degree with removing the element v of the current remaining set S that minimizes $f(v|S - v)$. We then get an ordering $v_1, v_2, \ldots, v_n$ of V where v_i is the ith element in the removal order.

Let $f : 2^V \to \mathbb{R}^{\geq 0}$ be a normalized, non-negative supermodular function. Let S^* be an optimal subset and let $\lambda^* = \frac{f(S^*)}{|S^*|}$. Furthermore, let v_j be the first element from the peeling order in the best suffix S_j. We want to prove a bound on the density of S_j with respect to λ^*. We begin with two useful lemmas.

Lemma 3.2. *For each $v \in S^*$, $f(v|S^* - v) \geq \lambda^*$.*

Proof. If $|S^*| = 1$, then v is the only element in S^* and $f(v|S^* - v) = f(S^*) - f(\emptyset) = \lambda^*$, by optimality of S^*. Suppose $|S^*| = 2$ and assume for contradiction that the claim is false. Then $f(v|S^* - v) < \lambda^*$ which implies $f(S^*) - f(S^* - v) < \frac{f(S^*)}{|S^*|}$. Rearranging, we get $f(S^* - v) > \frac{f(S^*)}{|S^*|} - f(S^*)$. Dividing by $|S^*| - 1$ on both sides and rearranging, we get $\frac{f(S^*-v)}{|S^*|-1} > \frac{|S^*|f(S^*)-f(S^*)}{|S^*|(|S^*|-1)}$ and factoring, we get $\frac{f(S^*-v)}{|S^*|-1} > \frac{f(S^*)}{|S^*|}$. But this implies that the set $S^* - v$ is a better set than S^*, which is a contradiction, completing the proof. □

Lemma 3.3. *Let v_j be the element in S^* that is peeled first, and consider the suffix $S_j = \{v_j, \ldots, v_n\}$. For $i \geq j$, we have $f(v_i|S_j - v_i) \geq f(v_j|S_j - v_j) \geq f(v_j|S^* - v_i) \geq \lambda^*$.*

Proof. We break this into three inequalities. The first inequality, $f(v_i|S_j - v_i) \geq f(v_j|S_j - v_j)$ follows from the peeling order of the algorithm, since the vertex with the lower marginal value gets peeled first. The second inequality, $f(v_j|S_j - v_j) \geq f(v_j|S^* - v_i)$ follows from supermodularity, since $S^* \subseteq S_j$. The last inequality, $f(v_j|S^* - v_i) \geq \lambda^*$, follows from Lemma 3.2. Chaining these together, we obtain the full inequality, completing the proof. □

We now prove a approximation guarantee for peeling based on the parameter $c_f = \max_{S \subseteq V} \frac{\sum_{v \in S} f(v|S-v)}{f(S)}$. From [7], we have the following theorem:

Theorem 3.1. *The greedy peeling algorithm has an approximation ratio of at least* $\frac{1}{c_f}$.

Proof. We want to prove a lower bound on the density of S_j. We note that

$$\frac{f(S_j)}{|S_j|} = \frac{f(S_j)}{|S_j|} \cdot \frac{\sum_{v \in S_j} f(v|S_j - v)}{\sum_{v \in S_j} f(v|S_j - v)} = \frac{f(S_j)}{\sum_{v \in S_j} f(v|S_j - v)} \cdot \frac{\sum_{v \in S_j} f(v|S_j - v)}{|S_j|}.$$

The fraction on the left is exactly $\frac{1}{c_f}$. Also, by Lemma 3.3, $f(v|S_j - v) \geq f(v_j|S_j - v_j)$ for any $v \in S_j$. Therefore we have

$$\frac{f(S_j)}{|S_j|} \geq \frac{1}{c_f} \cdot \frac{|S_j| f(v_j|S_j - v_j)}{|S_j|} = \frac{f(v_j|S_j - v_j)}{c_f}.$$

Applying Lemma 3.3 again, we have $f(v_j|S_j - v_j) \geq \lambda^*$, and so $\frac{f(S_j)}{|S_j|} \geq \frac{\lambda^*}{c_f}$, completing the proof. □

The preceding proofs of Lemma 3.2, Lemma 3.3, and Theorem 3.1 fill in the details of the proof in [7], which was itself adapted from [12]. The organization of this proof is inspired from the sketch of the proof given in the seminar talk by Chekuri [6].

The previous theorem means that obtaining an approximation bound on peeling immediately follows from obtaining (or estimating) an upper bound on c_f [7]. In fact, we can directly use this to obtain bounds on peeling for the DSP and for hypergraphs.

Corollary 3.1. *Peeling is a* $\frac{1}{2}$*-approximation for the DSP and a* $\frac{1}{r}$*-approximation for rank* r *hypergraphs.*

Proof. Recall that $c_f = \max_{S \subseteq V} \frac{\sum_{v \in S} f(v|S-v)}{f(S)}$. That is, c_f is the ratio of the maximum marginal values of elements in S divided by the valuation of S, maximized over all $S \subseteq V$. In graphs, the maximum marginal value of a vertex in $S \subset V$ is its degree in $G[S]$. But we know that in any graph, $\sum \deg(v) = 2|E|$. Therefore, for the DSP, $c_f = \max_{S \subseteq V} \frac{\sum_{v \in S} \deg_v(S)}{|E(S)|} = 2$. A similar argument works for hypergraphs; in an r-uniform hypergraph, $\sum \deg(v) = r|E|$, so in a general rank r hypergraph $c_f = \max_{S \subseteq V} \frac{\sum_{v \in S} \deg_v(S)}{|E(S)|} \leq r$. □

Chekuri et al. further generalize the analysis of greedy peeling for generalizations of hypergraphs called r-decomposable supermodular functions. This provides an even broader view of how greedy peeling behaves for general supermodular functions. For more details, see [7].

3.3 Iterative Peeling and Convergence for DSSP

In this section, we discuss the iterative peeling algorithm GREEDY++ for the DSP [4], its generalization SUPERGREEDY++ for the DSSP [7], and the proof by Chekuri et al. that SUPERGREEDY++ converges to a $(1 - \epsilon)$-approximation for the DSSP.

GREEDY++ is an extension of the Charikar's peeling algorithm that assigns iteratively assign loads to each element based on the previous peeling order and peels based on the weighted ordering of the vertices. After running for T iterations, it outputs a candidate for the best subset [4]. The algorithm can be described as follows:

1. Input: a graph G and a number of iterations T;
2. Keep an array of loads for each vertex, all initialized to 0. Let us denote the load of v during iteration i as $\text{load}_i(v)$;
3. In each iteration i, repeatedly find the vertex that minimizes $\text{current\_deg}(v) + \text{load}_{i-1}(v)$ and remove it (ties are broken arbitrarily);
4. Let $\text{load}_i(v) = \text{load}_{i-1}(v) + \text{current\_deg}(v)$ at the time of v's removal;

5. Let $v_{1,j}, v_{2,j}, \ldots, v_{n,j}$ be the ordering from the sequence of vertex removals during iteration j;
6. After T iterations, we output the suffix over all orderings $S_{i,j} = \{v_{i,j}, v_{i+1,j}, \ldots, v_{n,j}\}$ that induces the subgraph with the highest density λ.

Notice that the first iteration of this algorithm is identical to the procedure from Charikar's peeling algorithm. With each iteration of the algorithm, we obtain a new peeling order based on the previous iterations. This new order eventually stabilizes, so that typically the best suffix is of the form $S_{i,T} = \{v_{i,T}, v_{i+1,T}, \ldots, v_{n,T}\}$.

It is also interesting to that the iterative peeling algorithm is in some sense spreading out the load over the elements to minimize the maximum load over the elements, in some sort of aggregate sense [4, 7]. Since f is supermodular, as we peel away elements from S, the loads available for remaining elements to redistribute get smaller, and eventually they should converge to an optimal solution to the load balancing interpretation of the DSP given by Charikar [4, 5, 7].

In general, instead of sorting vertices by the induced degree of v in $G[S]$ where $S \subseteq V$, for general supermodular functions we can take the marginal value of v to the current set, i.e. $f(v|S - v)$. Everything else about iterative peeling remains the same. This variation of the iterative peeling algorithm for general supermodular functions was named SuperGreedy++ by Chekuri et al. [7].

Boob et al. ran experiments on several real-world graphs that seemed to indicate that Greedy++ always converges to a $(1 - \epsilon)$-approximation for the DSP, and conjectured that $T = O(\frac{1}{\epsilon^2})$ iterations are required [4]. Chekuri et al. proved the convergence of SuperGreedy++ for general supermodular functions:

Theorem 3.2. *Let $f : 2^V \to \mathbb{R}^{\geq 0}$ be a normalized, non-negative supermodular function and let $n = |V|$. SuperGreedy++ outputs a $(1-\epsilon)$-approximation after $T = O(\frac{\Delta_f \log n}{\lambda^* \epsilon^2})$ iterations, where λ^* is the optimal density and $\Delta_f = \max_{v \in V} f(V) - f(V - v)$.*

This also implicitly provides a proof of the convergence of Greedy++ for the DSP as a special case, albeit with a much weaker bound than conjectured by Boob et al.

Corollary 3.2. *Greedy++ outputs a $(1 - \epsilon)$-approximation after $T = O(\frac{\Delta \log n}{\lambda^* \epsilon^2})$ iterations, where λ^* is the optimal density and Δ is the maximum degree in the graph.*

A high level outline of how Chekuri et al. proved the convergence of SuperGreedy++ is as follows:

1. They defined a convex program for the DSSP:

$$\text{maximize } \hat{f} \text{ over } x \in \mathbb{R}^{\geq 0} \text{ s.t. } \sum_{v \in V} x_v \leq 1, \tag{3}$$

   where $\hat{f}$ denotes the Lovász extension of f. They view subsets of V as characteristic vectors $\mathbf{1}_S$. Taking the Lovász extension and maximizing over the whole vector space turns this maximization problem into a continuous, rather than a discrete problem, which makes it more tractable. They then show that this is an exact formulation of the DSSP.
2. Based on this convex program, and connnections between $\hat{f}$ and the symmetric group of V, they derive a new LP for the DSSP.
3. Solutions to the dual of this new LP can be rounded to solutions of the DSSP.
4. They describe an algorithm based on the multiplicative weights updates (MWU) framework that gives solutions to the dual LP.
5. They argue that SuperGreedy++ is a special case of the MWU framework and that it is sort of approximately rounding the solution provided by MWU. The main argument here is that peeling orderings from SuperGreedy++ approximately correspond to the reverse of orderings selected in the MWU algorithm. Therefore, increases in loads in SuperGreedy++ can more or less be mapped to increases in weights in MWU.
6. The bounds they obtain are by proxy to the MWU algorithm, which they argued that SuperGreedy++ roughly approximates.

We note a few interesting things from this proof. First, Charikar's LP for the DSP [5] is a special case of (3), which is the convex program for the DSSP described by Chekuri et al. [7]. Secondly, the main argument here is that SuperGreedy++ is actually a rough approximation for an MWU algorithm that solves the DSSP, so this proof is somewhat indirect and by proxy. Lastly, since the DSP is a special case of the DSSP, this is also a proof of convergence of SuperGreedy++, albeit a very indirect one.

4 Conclusion

In this report, we have looked at reasons to study dense subgraph discovery techniques, with a particular focus on the densest subgraph problem (DSP). We have also covered a few rudiments of the theory of submodularity (and by extension, supermodularity), and studied the DSP from the purview of the more generalized densest supermodular subset problem (DSSP). We have shown how approximation algorithms for the DSP are also approximations for the DSSP and generalized their analysis for all supermodular functions.

It is interesting to see how taking a more generalized outlook on a problem can provide deeper insights on the relationships between problems (sometimes showing that they are, in fact, the same problem) and on the methods used to solve such problems. This is exactly what the work of Chekuri et al. has accomplished: viewing the DSP through the lens of supermodularity provides deeper insights into how and why the peeling procedure works and suggests that such a procedure could perhaps be leveraged for other related problems.

It is also of personal interest to note that this work explains why implementations of SuperGreedy++ are experimentally slower on some hypergraphs than others. If we think of graphs as modelling pairwise correlations between elements, and rank r hypergraphs as modelling correlations between groups with at most r elements, it becomes clear that the more elements are grouped together in a single "unit of correlation", the worse the performance of peeling becomes. This also suggests that as the average cardinality of edges in a hypergraph decreases, the overall efficiency of SuperGreedy++ for finding dense subgraphs decreases, and so SuperGreedy++ works best on hypergraphs of low rank. This validates experimental findings from prior work I have done. While it has been proven that iterative peeling will eventually converge to an optimal subset for any supermodular function, iterative peeling is perhaps unideal for general r-decomposable supermodular functions, especially those with a high rank r.

Lastly, I find the proof of convergence of the DSSP to be interesting because it is quite indirect; the natural phrasing of the DSSP as a convex program is first converted to a linear program, which is then solved using a MWU algorithm, and then an argument is made that SuperGreedy++ is in fact closely approximating the MWU algorithm, which is why it converges. The original conjecture of Boob et al. is still an open question, and in a seminar talk [6] about [7], Chekuri states that he believes that the original conjecture is perhaps too bold. However, I wonder if a somewhat tighter bound on the number of iterations could be obtained using a more direct approach to the proof. Perhaps this could be investigated in future work.

References

[1] Yuichi Asahiro, Kazuo Iwama, Hisao Tamaki, and Takeshi Tokuyama. 1996. Greedily finding a dense subgraph. In *Algorithm Theory — SWAT'96*. Rolf Karlsson and Andrzej Lingas, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 136–148. ISBN: 978-3-540-68529-6. DOI: 10.1007/3-540-61422-2_127.

[2] Jeff A. Bilmes. 2016. Deep mathematical properties of submodularity with applications to machine learning. (2016). https://www.youtube.com/watch?v=ZycBUGLD22E.

[3] Jeff A. Bilmes. 2020. Submodular functions, optimization, and applications to machine learning. (2020). https://people.ece.uw.edu/bilmes/classes/ee563/ee563_fall_2020/.

[4] Digvijay Boob, Yu Gao, Richard Peng, Saurabh Sawlani, Charalampos Tsourakakis, Di Wang, and Junxing Wang. 2020. Flowless: extracting densest subgraphs without flow computations. In *Proceedings of The Web Conference 2020* (WWW '20). Association for Computing Machinery, Taipei, Taiwan, 573–583. ISBN: 9781450370233. DOI: 10.1145/3366423.3380140.

[5] Moses Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *Approximation Algorithms for Combinatorial Optimization*. Klaus Jansen and Samir Khuller, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 84–95. ISBN: 978-3-540-44436-7. DOI: 10.1007/3-540-44436-X_10.

[6] Chandra Chekuri. 2022. Densest subgraph – supermodularity, iterative peeling, and flow. (2022). https://www.youtube.com/watch?v=DO8iNCwmp0c.

[7] Chandra Chekuri, Kent Quanrud, and Manuel R. Torres. 2022. Densest subgraph: supermodularity, iterative peeling, and flow. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1531–1555. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611977073.64. DOI: 10.1137/1.9781611977073.64.

[8] Jie Chen and Yousef Saad. 2012. Dense subgraph extraction with application to community detection. *IEEE Transactions on Knowledge and Data Engineering*, 24, 7, 1216–1230. DOI: 10.1109/TKDE.2010.271.

[9] Steven R. Corman, Timothy Kuhn, Robert D. Mcphee, and Kevin J. Dooley. 2002. Studying complex discursive systems. *Human Communication Research*, 28, 2, 157–206. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1468-2958.2002.tb00802.x. DOI: https://doi.org/10.1111/j.1468-2958.2002.tb00802.x.

[10] Giorgio Gallo, Michael D. Grigoriadis, and Robert E. Tarjan. 1989. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18, 1, 30–55. eprint: https://doi.org/10.1137/0218003. DOI: 10.1137/0218003.

[11] A. V. Goldberg. 1984. Finding a Maximum Density Subgraph. Tech. rep. USA. DOI: 10.5555/894477.

[12] Samir Khuller and Barna Saha. 2009. On finding dense subgraphs. In *Automata, Languages and Programming*. Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 597–608.

[13] Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM*, 46, 5, (Sept. 1999), 604–632. DOI: 10.1145/324133.324140.

[14] Tommaso Lanciano, Atsushi Miyauchi, Adriano Fazzone, and Francesco Bonchi. 2024. A survey on the densest subgraph problem and its variants. *ACM Computing Surveys*, 56, 8, (Apr. 2024), 1–40. DOI: 10.1145/3653298.

[15] László Lovász. 1983. Submodular functions and convexity. In *Mathematical Programming The State of the Art: Bonn 1982*. Achim Bachem, Bernhard Korte, and Martin Grötschel, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 235–257. ISBN: 978-3-642-68874-4. DOI: 10.1007/978-3-642-68874-4_10.

[16] S. Thomas McCormick. 2005. Submodular function minimization. In *Discrete Optimization*. Handbooks in Operations Research and Management Science. Vol. 12. K. Aardal, G.L. Nemhauser, and R. Weismantel, editors. Elsevier, 321–391. DOI: https://doi.org/10.1016/S0927-0507(05)12007-6.

[17] Jean-Claude Picard and Maurice Queyranne. 1982. A network flow solution to some nonlinear 0-1 programming problems, with applications to graph theory. *Networks*, 12, 2, 141–159. DOI: https://doi.org/10.1002/net.3230120206.

[18] Charalampos E. Tsourakakis and Tianyi Chen. Dense subgraph discovery (DSD) theory and applications. SIAM International Conference on Data Mining (SDM21), (2021). https://tsourakakis.com/dense-subgraph-discovery-theory-and-applications-tutorial-sdm-2021/.

[19] Jan Vondrák. 2010. Continuous extensions of submodular functions. (2010). https://theory.stanford.edu/~jvondrak/CS369P/lec17.pdf.