Densest Subgraph: Supermodularity, and Iterative Peeling

Chandra Chekuri

Univ. of Illinois, Urbana-Champaign

Based on joint works with

Harb ElFarouk, Kent Quanrud and Manuel Torres

UBC, June 13, 2024

Densest Subgraph (DSG)

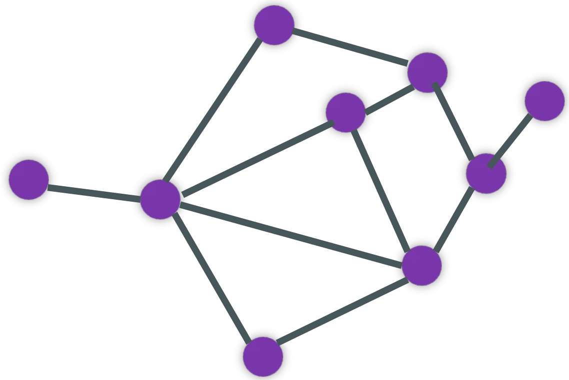
$G = (V, E)$ undirected graph

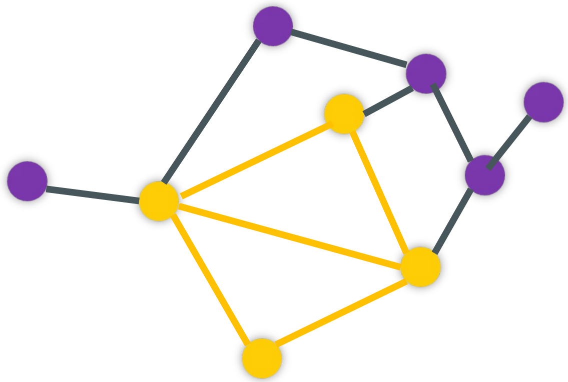
Find "dense" subgraph(s)

$$density(S) = \frac{|E(S)|}{|S|}$$

$$\lambda^* = \max_{S \subseteq V} \frac{|E(S)|}{|S|}$$

Example





$$\lambda^* = \frac{5}{4}$$

Dense Subgraph Discovery

$$density(S) = \frac{f(S)}{|S|}$$

- Triangle density: f(S) = # of triangles in G[S] [Tsourakakis'14]
- k-clique density: f(S) = # of k-cliques in G[S] [Tsourakakis'15]
- Hypergraphs: f(S) = # of hyperedges in G[S] [folklore?]
- p-mean density: $f(S) = \sum_{v \in S} \deg(v, S)^p$ [Benson-Kleinberg-Veldt'21]
- **Constrained versions:** [many authors]

  $$\max f(S) \; s.t \; |S| = k, |S| \leq k, |S| \geq k$$
- Directed graph version: [Kannan-Vinay'99,Charikar'00]

Polynomial Solvability

DSG is poly-time solvable

- Reduction to flow [Picard-Queyranne'82, Goldberg'84]
- Reduction to submodular function minimization [folklore]
- LP relaxation [Charikar'00]

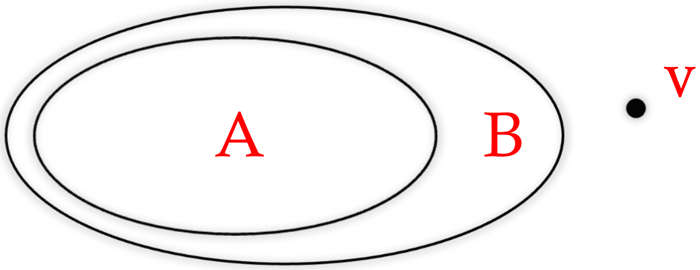
Sub and Supermodularity

Real-valued set function $f: 2^V \rightarrow R$ is **submodular** if

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \quad \forall A, B$$

Equivalently:

$$f(A + v) - f(A) \geq f(B + v) - f(B) \quad A \subset B, v \notin \mathrm{B}$$



Sub and Supermodularity

$f: 2^V \rightarrow R$ is **supermodular** iff $-f$ is **submodular**

$$f(A) + f(B) \leq f(A \cap B) + f(A \cup B) \quad \forall A, B$$

Marginal value: $f(v \mid S) = f(S + v) - f(S)$

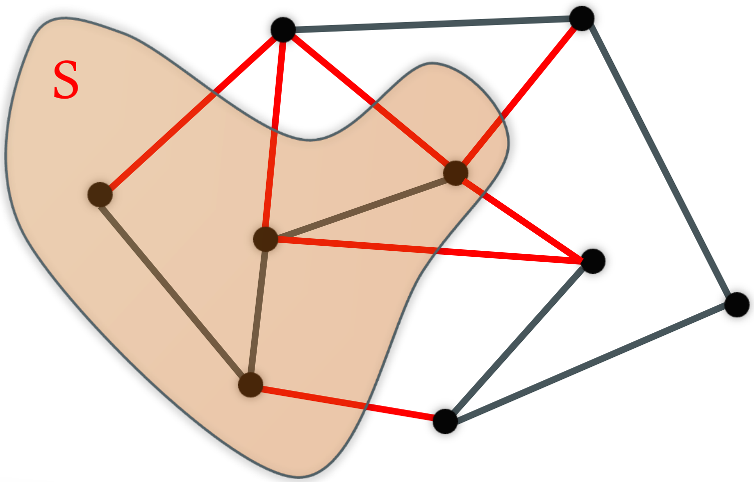
Supermodular:

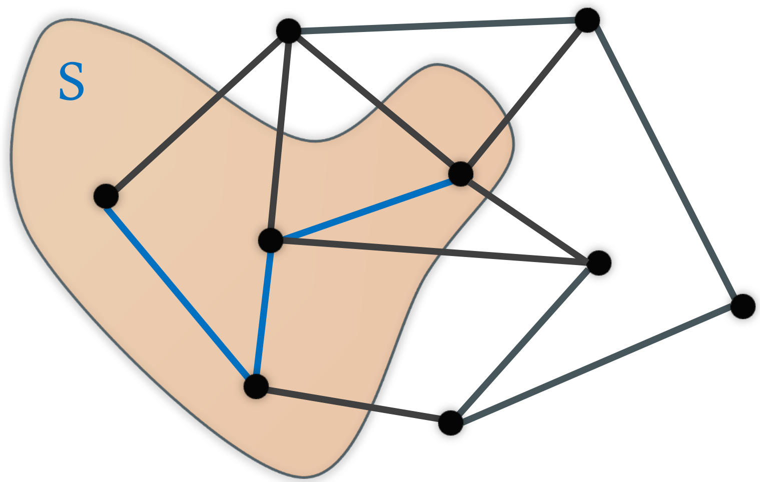
$$f(v \mid B) \geq f(v \mid A) \qquad A \subset B, v \in B - A$$

Sub and Supermodularity

Given graph $G = (V, E)$

- $f(S) = |\delta(S)|$ is submodular and non-neg
- $f(S) = |E(S)| = \frac{1}{2}(\sum_v \deg(v) - |\delta(S)|)$ is supermodular, non-negative and monotone





Densest Supermodular Set (DSS)

Given supermodular $f: 2^V \to R_+$ find $\max_S \frac{f(S)}{|S|}$

Decision version: check if $\exists S \; s.t \; \frac{f(S)}{|S|} \geq \lambda$

Check if $\exists S \; s.t \; \lambda|S| - f(S) \leq 0$

Poly-time via submodular function minimization

Some Recent Directions on Densest Subgraph Discovery

- Fast *approximate* algorithms for *(very) large* graphs
- Variations in objective and applications
- Streaming (approximate) algorithms
- Parallel (approximate) algorithms
- Dynamic (approximate) algorithms
- …

My Motivation

- Conjecture of [Boob-Gao-Peng-Sawlani-Tsourkakis-Wang-Wang'20] on a simple iterative greedy alg.
- Faster approximations for mixed packing and covering LPs (DSG is a special case)
- Connections to supermodularity
- Discrete + continuous

Results at high-level

- **Fast approximate algorithm:** $(1-\epsilon)$ approximation for densest subgraph in $O\left(m\frac{polylog(n)}{\epsilon}\right)$ time
- **Affirmative answer to conjecture** of [Boob et al]
- **Generalization to supermodular functions**
- Other results …

Connections *which are simple in retrospect but helpful for both theory and practice*

Papers

- *Densest Subgraph: Supermodularity, Iterative Peeling, and Flow* [CQT SODA'22]
- *Faster and Scalable Algorithms for Densest Subgraph and Decomposition* [HQC NeuRIPS'22]
- *(1- ϵ)-approximate fully dynamic densest subgraph: linear space and faster update time* [CQ'22/23]
- *Convergence to Lexicographically Optimal Base in a (Contra)Polymatroid and Applications to Densest Subgraph and Tree Packing* [HQC '23]
- *On the Generalized Mean Densest Subgraph Problem: Complexity and Algorithms* [CT'23]

Rest of the talk

- Charikar's LP Relaxation
- Peeling and Iterative Peeling
- Connections and ideas about proof of convergence

Charikar's LP Relaxation

Integer Programming Formulation

$$\max \frac{\sum_{uv\in E} x_{uv}}{\sum_v z_v}$$

$$x_{uv} \leq \min(z_u, z_v) \quad uv \in E$$

$$x, z \; binary \; variables$$

$$\max_S \frac{|E(S)|}{|S|}$$

$z_v \in \{0,1\}$ $v \in S$?

$x_{uv} \in \{0,1\}$ $uv \in E(S)$?

Charikar's LP Relaxation

$$\max \sum_{uv \in E} x_{uv}$$

$$\sum_v z_v = 1$$

$$x_{uv} \le \min(z_u, z_v) \quad uv \in E$$

$$x, z \ge 0$$

$$\max \frac{|E(S)|}{|S|}$$

$z_v \in \{0,1\}$ $v \in S$?

$x_{uv} \in \{0,1\}$ $uv \in E(S)$?

Theorem: [Charikar'00] LP is optimal for DSG

Charikar's LP Relaxation

Primal

$$\max \sum_{uv \in E} x_{uv}$$

$$\sum_v z_v = 1$$

$$x_{uv} \leq \min(z_u, z_v) \quad uv \in E$$

$$x, z \geq 0$$

Dual

$$\min D$$

$$y_{uv,u} + y_{uv,v} \geq 1 \quad uv \in E$$

$$\sum_{uv \in E} y_{uv,v} \leq D \quad v \in V$$

$$y \geq 0$$

Theorem: [Charikar'00] LP is optimal for DSG

Flow Reduction via Dual

Observed in [Boob et al]

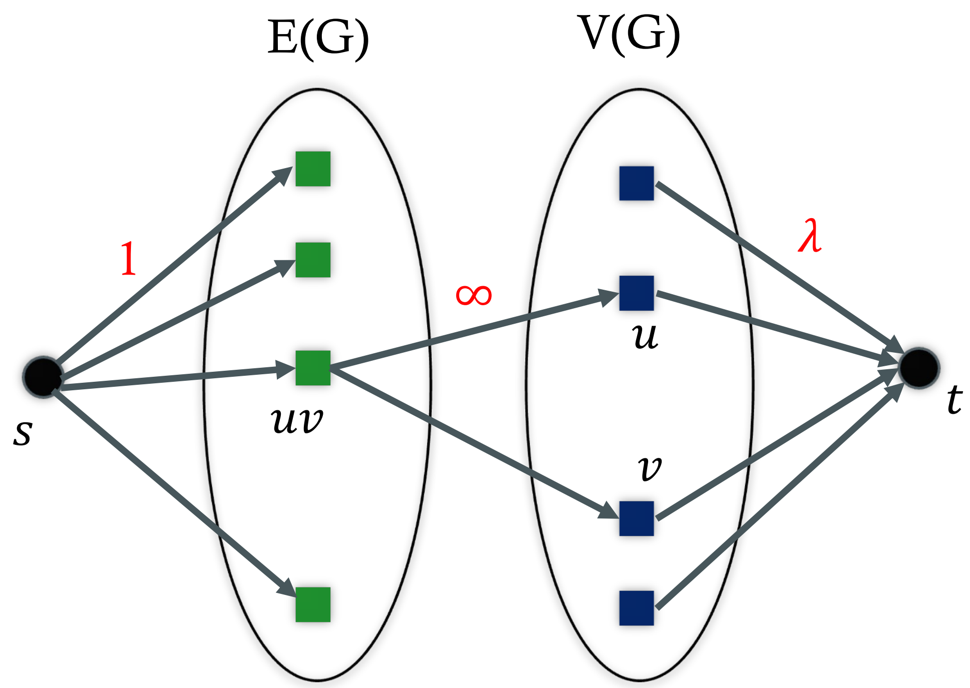
$\min D$

$y_{uv,u} + y_{uv,v} \geq 1 \quad uv \in E$

$\sum_{uv \in E} y_{uv,v} \leq D \quad v \in V$

$y \geq 0$

Fractional perfect matching



Flow network H_λ

Claim: Max-flow in $H_\lambda = |E|$ iff $\lambda \geq \lambda^*$

Utility of LP

- Dual LP can be viewed as a flow problem --- simpler formulation than [Goldberg,Picard-Queyranne]. Dual LP computes *fractional arboricity*. $\lambda^* = fractional\ arboricity$
- Dual LP is mixed-packing and covering LP. Hence can solve via approximate methods [Bahmani-Goel-Munagala'14] [Boob-Sawlani-Wang'19]
- More connections soon

Flow based Approx Algorithm

[CQT'22]

Theorem: $(1-\epsilon)$ approximation for DSG in $O\left(m\frac{polylog(n)}{\epsilon}\right)$ time via *approximate* flow

Improvement: $\frac{1}{\epsilon}$ $instead\ of$ $\frac{1}{\epsilon^2}$

Key structural idea: *short* $(length \leq c \log n /\epsilon)$ augmenting paths suffice to get $(1-\epsilon)$ approximation

Empirical utility of idea not yet unexplored

Peeling Algorithm

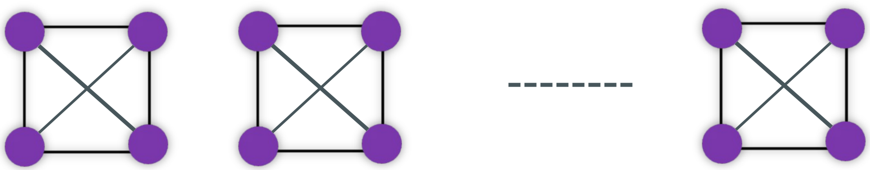
[Asahiro etal 00, Charikar 00]

- For $i = 1\ to\ n$ do
  - v_i is in *min-degree* vertex in G
  - $G \leftarrow G\ - v_i$

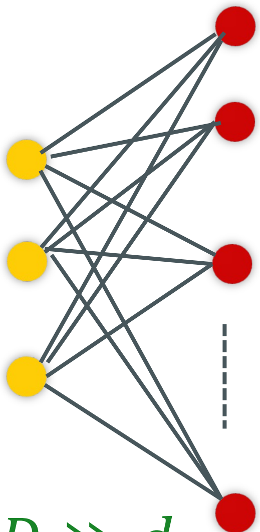
- $v_1, v_2, \ldots, v_n$ is *ordering* created by algorithm
- $S_i \leftarrow \{v_i, v_{i+1}, \ldots, v_n\}$
- Output $argmax_i \frac{|E(S_i)|}{|S_i|}$

Theorem: [Charikar'00] Greedy peeling is a ½ approximation for DSG (proof via LP)

(Tight) Example



K_{d+1}

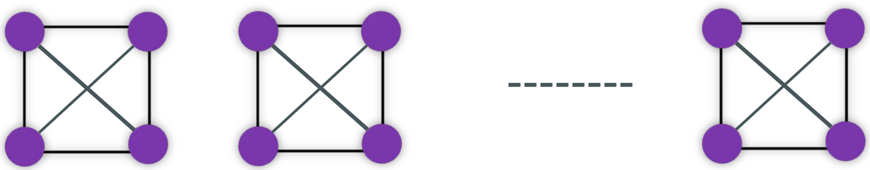


$K_{d,D}$ $D \gg d$

$\lambda^* \simeq d$ via $K_{d,D}$

$\lambda(G) \simeq \frac{d}{2}$

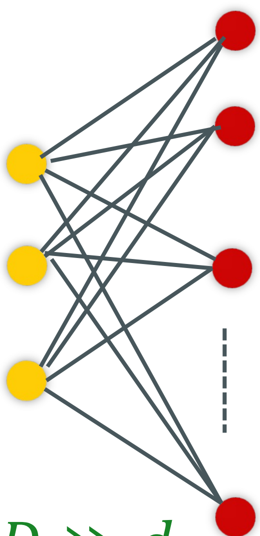
(Tight) Example



K_{d+1}

$\lambda^* \simeq d$ via $K_{d,D}$

$\lambda(G) \simeq \frac{d}{2}$



$K_{d,D}$ $D \gg d$

Peeling order

Peeling and DSS

Given supermodular function $f\colon\ 2^V \rightarrow R_+$

- For $i = 1\ to\ n$ do
  - $v_i \leftarrow argmin_v\, f(v \mid V - v)$
  - $V \leftarrow V - v_i$
  - Restrict f to $V - v_i$

- $v_1, v_2, \ldots, v_n$ is ordering created by algorithm
- $S_i \leftarrow \{v_i, v_{i+1}, \ldots, v_n\}$
- Output $argmax_i\, \frac{|f(S_i)|}{|S_i|}$

Peeling and DSS

Question: How can we characterize for general f?

$$c_f = \max_S \frac{\sum_{v \in S} f(v \mid S - v)}{f(S)}$$

Supermodularity:

$$\sum_{v \in S} f(v \mid S - v) \geq f(S) \Rightarrow c_f \geq 1$$

Peeling and DSS

$$c_f = \max_S \frac{\sum_{v \in S} f(v \mid S - v)}{f(S)}$$

Theorem: Peeling is a $\frac{1}{c_f}$ approximation for DSS

Proof is a simple adaptation of the combinatorial proof for DSG [Khuller-Saha'09]

Can also do it via relaxation ala [Charikar'00]

Peeling and DSS

Theorem: Peeling is a $\frac{1}{c_f}$ approximation for DSS

- Graphs: $c_f = \max_S \frac{\sum_{v \in S} \deg(v,S)}{|E(S)|} = 2$
- Hypergraphs: $c_f = r$ where r is rank
- p-th mean in graphs: $c_f = p + 1$

Unifies all the known bounds on greedy peeling

Iterative Peeling

[BGPSTWW'20]

- Heuristic inspired by Dual-LP and MWU
- Goal: improve ½ approx to $(1 - \epsilon)$ approx.
- Peel several times by adjusting "load"
- Creates a new ordering in each iteration
- Pick best suffix among all orderings

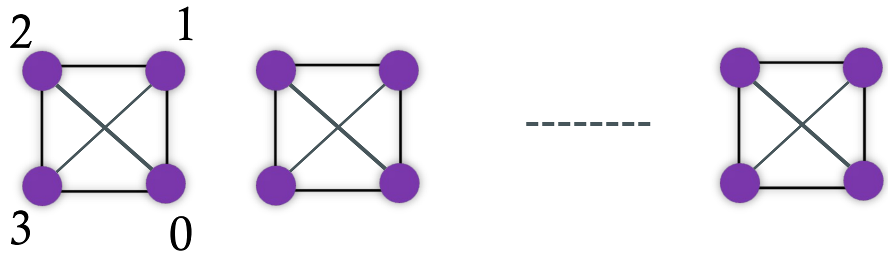
Iterative Peeling

[BGPSTWW'20]

Greedy++

- $load(v, 0) = 0 \quad for\ all\ v$
- For $t = 1\ to\ T$ do
    - $G' \leftarrow G$
    - For $i = 1\ to\ n$ do
        - $v_{t,i} \leftarrow argmin_v \deg(v) + load(v, t - 1)$
        - $load(v_{t,i}, t) = load(v_{t,i}, t - 1) + \deg(v_{t,i})$
        - $G' \leftarrow G' - v_{i,t}$
- $S_{t,i} \leftarrow \{v_{t,i}, \dots, v_{t,n}\}$
- Output $argmax_{i,t} \frac{|E(S_{t,i})|}{|S_{t,i}|}$

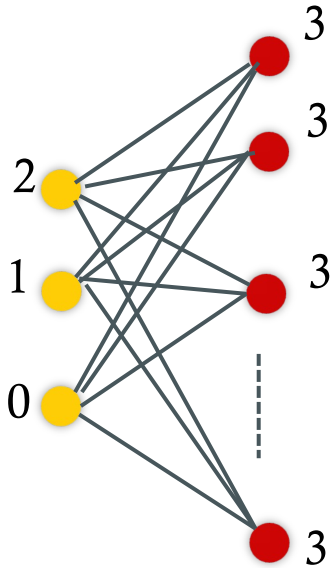
Example



K_{d+1}

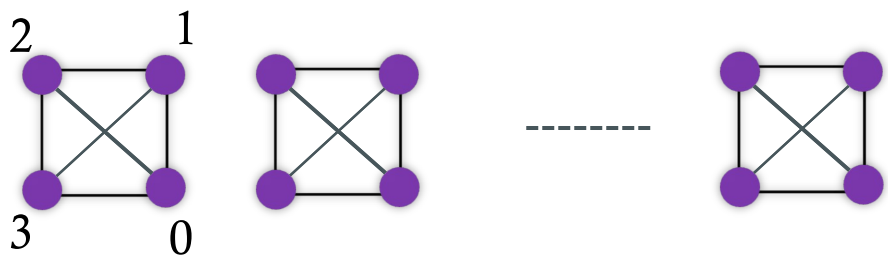
$\lambda^* \simeq d$

Peeling: $\lambda \simeq \frac{d}{2}$



$K_{d,D}$ $D \gg d$

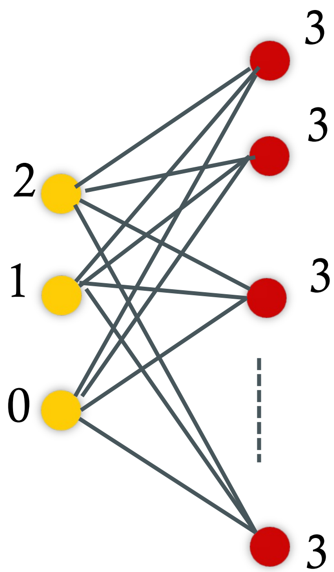
Example

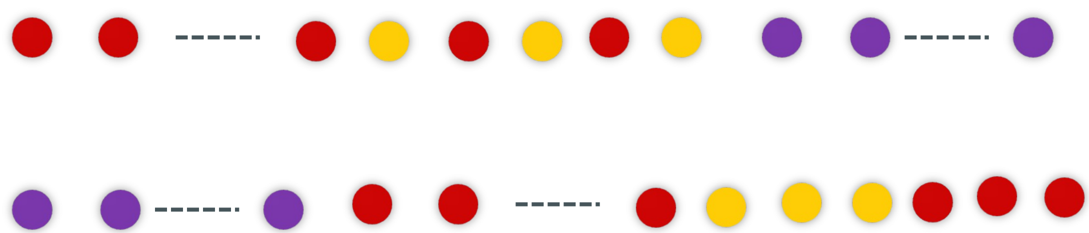


K_{d+1}

$\lambda^* \simeq d$

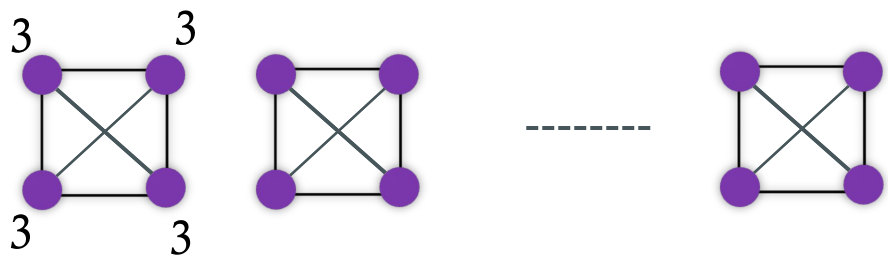
Peeling: $\lambda \simeq \frac{d}{2}$





$K_{d,D}$ $D \gg d$

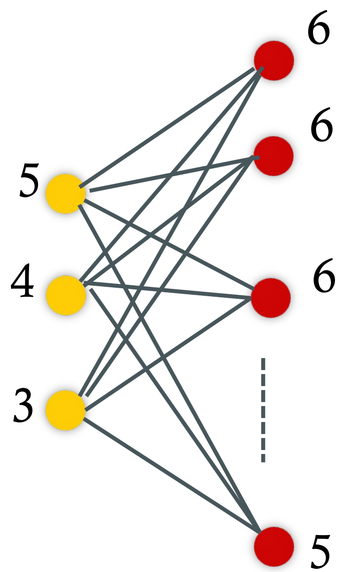
Example

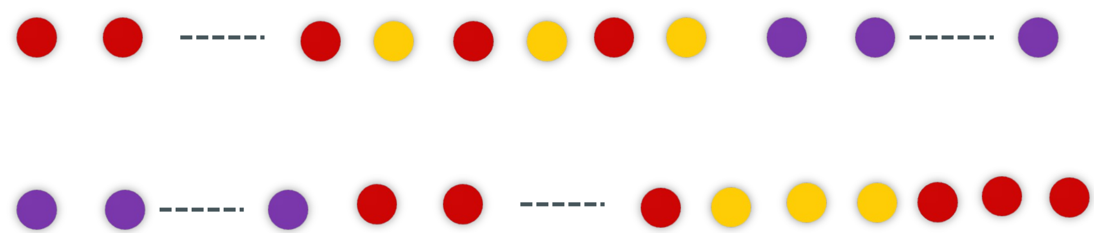


K_{d+1}

$\lambda^* \simeq d$

Peeling: $\lambda \simeq \frac{d}{2}$





$K_{d,D}$ $D \gg d$

Conjecture

[BGPSTWW'20]

Conjecture: Greedy++ is a $(1 - \epsilon)$ approximation after $O\left(\frac{1}{\epsilon^2}\right)$ iterations for DSG

Seems to work very well in practice. Implementation runs very fast even on large graphs and converges quickly on many real-world graphs

Iterative Peeling for DSS

Given supermodular $f: 2^V \to R_+$ find $\max_S \frac{f(S)}{|S|}$

SuperGreedy++

- $load(v, 0) = 0 \quad for\ all\ v$
- For $t = 1\ to\ T$ do
  - $S_{t,0} \leftarrow V$
  - For $i = 1\ to\ n$ do
    - $v_{t,i} \leftarrow argmin_{v \in S_{t,i}} f(v|S_{t,i} - v) + load(v, t-1)$
    - $load(v_{t,i}, t) = load(v_{t,i}, t-1) + f(v_{t,i}|S_{t,i} - v_{t,i})$
    - $S_{t,i+1} \leftarrow S_{t,i} - v_{t,i}$
- Output $argmax_{t,i} \frac{|f(S_{t,i})|}{|S_{t,i}|}$

Iterative Peeling for DSS

[CQT'22]

Theorem: SuperGreedy++ converges to a $(1 - \epsilon)$ approximation in $O(\frac{1}{\epsilon^2} \frac{\max_v f(v)}{\lambda^*} \log n)$ iterations

Corollary: Greedy++ converges to a $(1 - \epsilon)$ approximation for DSG in $O(\frac{1}{\epsilon^2} \frac{\Delta(G)}{\lambda^*} \log n)$ iterations

Proof Idea

- Express DSS as an LP relaxation
  - Generalize Charikar's LP for DSG via Lovasz-extension of supermodular/submodular functions
  - Rewrite as LP via an *ordering based* view of Lovasz-extension
- Relate SuperGreedy++ iterations to a multiplicative-weight update (MWU) algorithm via LP
  - SuperGreedy++ iterations are ***not*** MWU iterations but can show ***approximate*** relationship which is the main technical part

Different Perspective/Proof

[HQC'22, 23]

- Focus on DSS
- Make connection to principal partition of sub/supermodular function
- Fujishige's result on lexicographically optimal base in a polymatroid
- Frank-Wolfe method and convergence analysis

Dense Subgraph Decomposition

Lemma: There is a **unique maximal** ”densest” subgraph in any graph G

Suppose A and B are *maximal* sets with density λ^*

1. $f(A) + f(B) \leq f(A \cup B) + f(A \cap B)$
2. $|A| + |B| = |A \cup B| + |A \cap B|$

Implies $A \cup B$ has density λ^*

Dense Decomposition

Fix supermodular function $f: 2^V \rightarrow R_+$ (monotone, non-negative)

S_1 is unique maximal densest set for f with density λ_1

Function $f_{S_1}: 2^{V-S_1} \rightarrow R_+$ obtained by *contracting* S_1 (also supermodular)

S_2 is unique maximal densest set for f_{S_1} with density λ_2

Observation: $\lambda_1 > \lambda_2$

Dense Subgraph Decomposition

Fix supermodular function $f: 2^V \to R_+$ (monotone, non-negative)

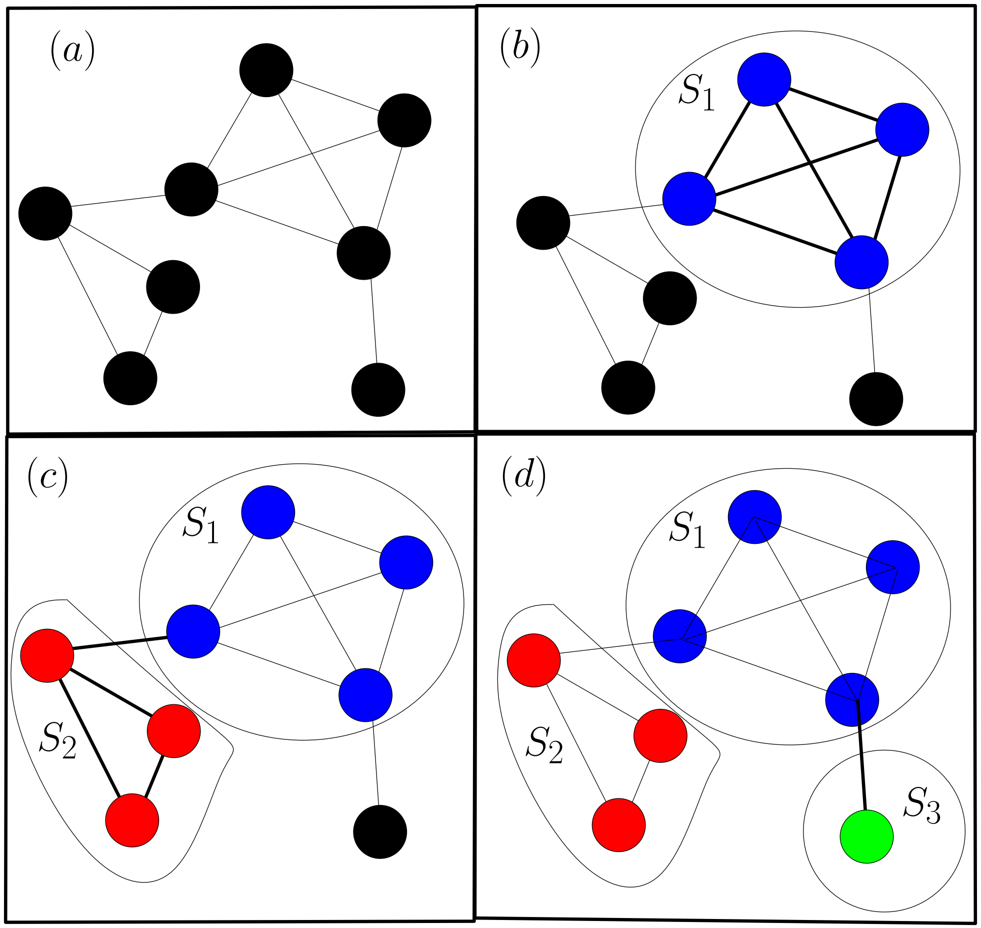
Can partition V into $S_1, S_2, \ldots, S_k$ with decreasing densities $\lambda_1 > \lambda_2 > \cdots > \lambda_k$

Called the dense (subgraph) decomposition

For each $v \in S_i$ associate $\lambda(v) = \lambda_i$

$\bar{\lambda} \in R^V$ the *dense decomposition vector*

Dense Subgraph Decomposition



Dense Decomposition

Fix supermodular function $f: 2^V \rightarrow R_+$ (monotone, non-negative)

Alternatively: consider

$$\max f(S) - \lambda|S| \text{ as } \lambda \text{ varies from } -\infty \text{ to } \infty$$

Optimum changes only a finite number of times corresponding to *nested family* of sets: $S_1, S_1 \cup S_2, S_1 \cup S_2 \cup S_3, \ldots, V = S_1 \cup S_2 \ldots \cup S_k$

Well-studied in graph/matroid/submodular literature: survey by Fujishige *"Theory of Principal Partitions Revisited"*

Computing Dense Decomposition Vector

[Fujishige'80]

Theorem: Dense decomposition vector $\bar{\lambda}$ is the *unique lexicographically minimal base* in the contra polymatroid associated with f

Fujishige considered submodular functions but can be easily adapted to supermodular functions

Polymatroid

Suppose $g: 2^V \rightarrow R_+$ is a monotone *submodular* function such that $g(\emptyset) = 0$ (normalized)

[Edmonds] *polymatroid* associated with g is the polytope in R^{n} (here $n = |V|$)

$$
\begin{aligned}
x(S) &\le f(S) \quad && \text{for all } S \subseteq V \\
x_v &\ge 0 \quad && \text{for all } v \in V
\end{aligned}
$$

Contra Polymatroid

Suppose $f: 2^V \to R_+$ is a monotone *supermodular* function such that $f(\emptyset) = 0$ (normalized)

Contra polymatroid associated with f is the polytope in R^{n} (here $n = |V|$)

$$x(S) \geq f(S) \quad for\ all\ S \subseteq V$$
$$x_v \geq 0 \quad for\ all\ v \in V$$

Base Contra Polymatroid

Suppose $f: 2^V \to R_+$ is a monotone *supermodular* function such that $f(\emptyset) = 0$ (normalized)

Base Contra polymatroid associated with f is the polytope

$$
\begin{aligned}
x(S) &\geq f(S) \quad && for\ all\ S \subseteq V \\
x(V) &= f(V) && \\
x_v &\geq 0 && for\ all\ v \in V
\end{aligned}
$$

Each vector $y \in B_f$ is a **base** of f

Lexicographically optimal base & Dense Decomposition

[Fujishige'80] (interpreted/paraphrased)

Theorem: $f: 2^V \rightarrow R_+$ is a monotone *supermodular* function and let B_f be its base contra polymatroid. Then there is a unique lexicographically minimum base y^* and

1. $y^* = \bar{\lambda}$
2. max density $\lambda_1 = \min \max_{v} x_v \; s.t \; x \in B_f$ (an LP)
3. y^* is the unique opt solution to *quadratic* program

$$\min \sum_v x_v^2 \quad s.t \; x \in B_f$$

Back to DSG

Recall for densest subgraph: $f(S) = |E(S)|$

What is Fujishige's "relaxation"?

Variable x_v for each vertex $v \in V$

$$
\begin{aligned}
& \min D \\
& x_v \leq D \quad for\ all\ v \in V \\
& \textstyle\sum_v x_v = m \\
& \textstyle\sum_{v \in S} x_v \geq |E(S)| \quad for\ all\ S \subseteq V \\
& x_v \geq 0 \ for\ all\ v \in V
\end{aligned}
$$

Back to DSG

[HQC'22]

Question: How is this exponential sized LP related to Charikar's LP?

- Dual of Charikar's LP is "equivalent" to Fujishige's relaxation!
- Charikar's LP can be viewed as a compact extended formulation that is specific to DSG
- Charikar's primal LP can be recast via the Lovasz extension of supermodular function

Frank-Wolfe for solving QP

Optimum solution to quadratic program:

$$\min \sum_{v} x_v^2 \quad \text{such that } x \in B_f$$

is the dense decomposition vector

How do we solve this quadratic program?

Frank-Wolfe from convex optimization is ideal because *linear optimization* over B_f is easy: greedy algorithm is optimal for polymatroid/contra polymatroids [Edmonds]

Frank-Wolfe for solving QP

Optimum solution to quadratic program:

$$\min \sum_{v} x_v^2 \quad \text{such that } x \in B_f$$

is the dense decomposition vector

How do we solve this quadratic program?

Frank-Wolfe from convex optimization is ideal because *linear optimization* over B_f is easy: greedy algorithm is optimal for polymatroid/contra polymatroids [Edmonds]

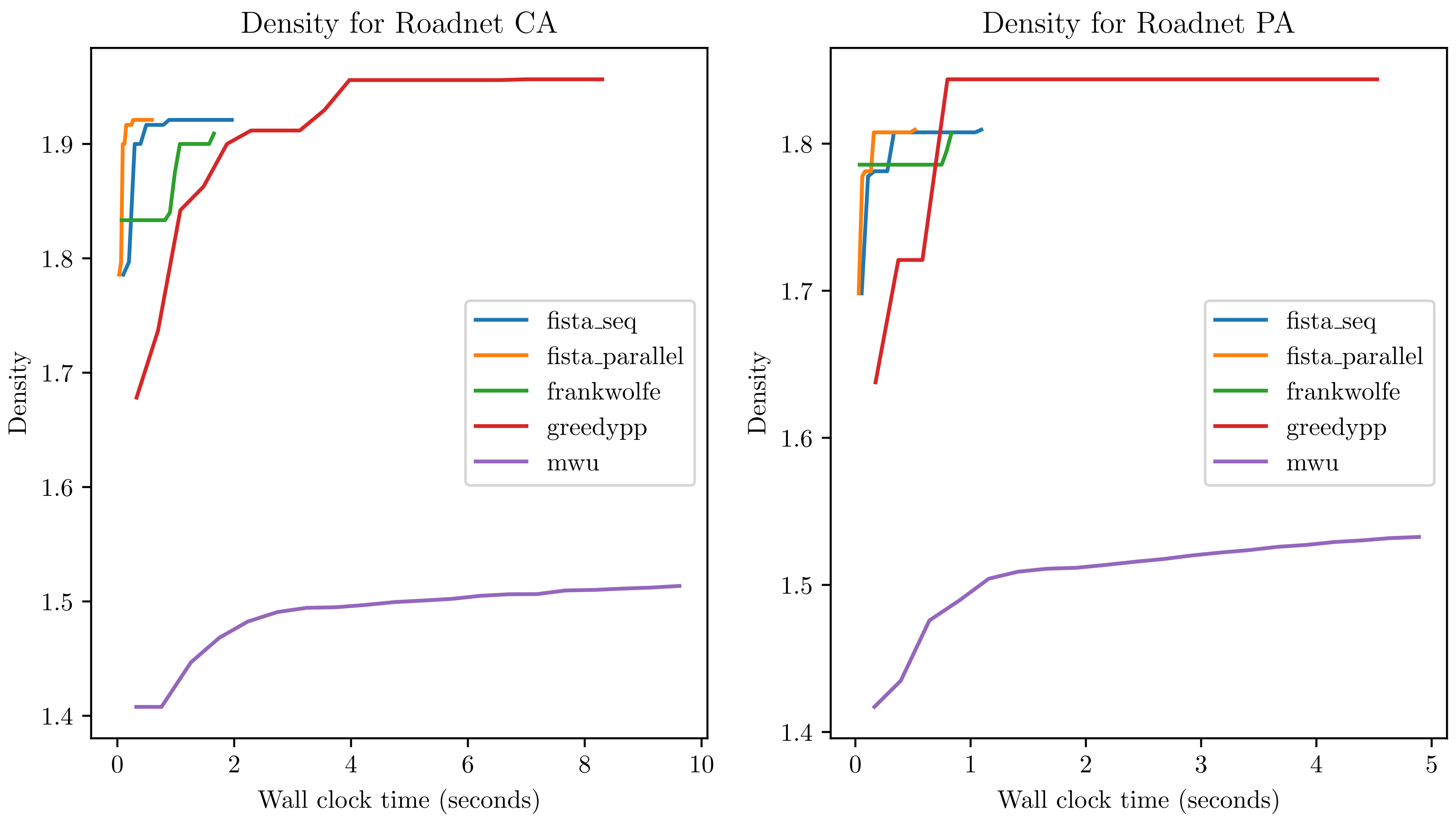
Back to Greedy++ and SuperGreedy++

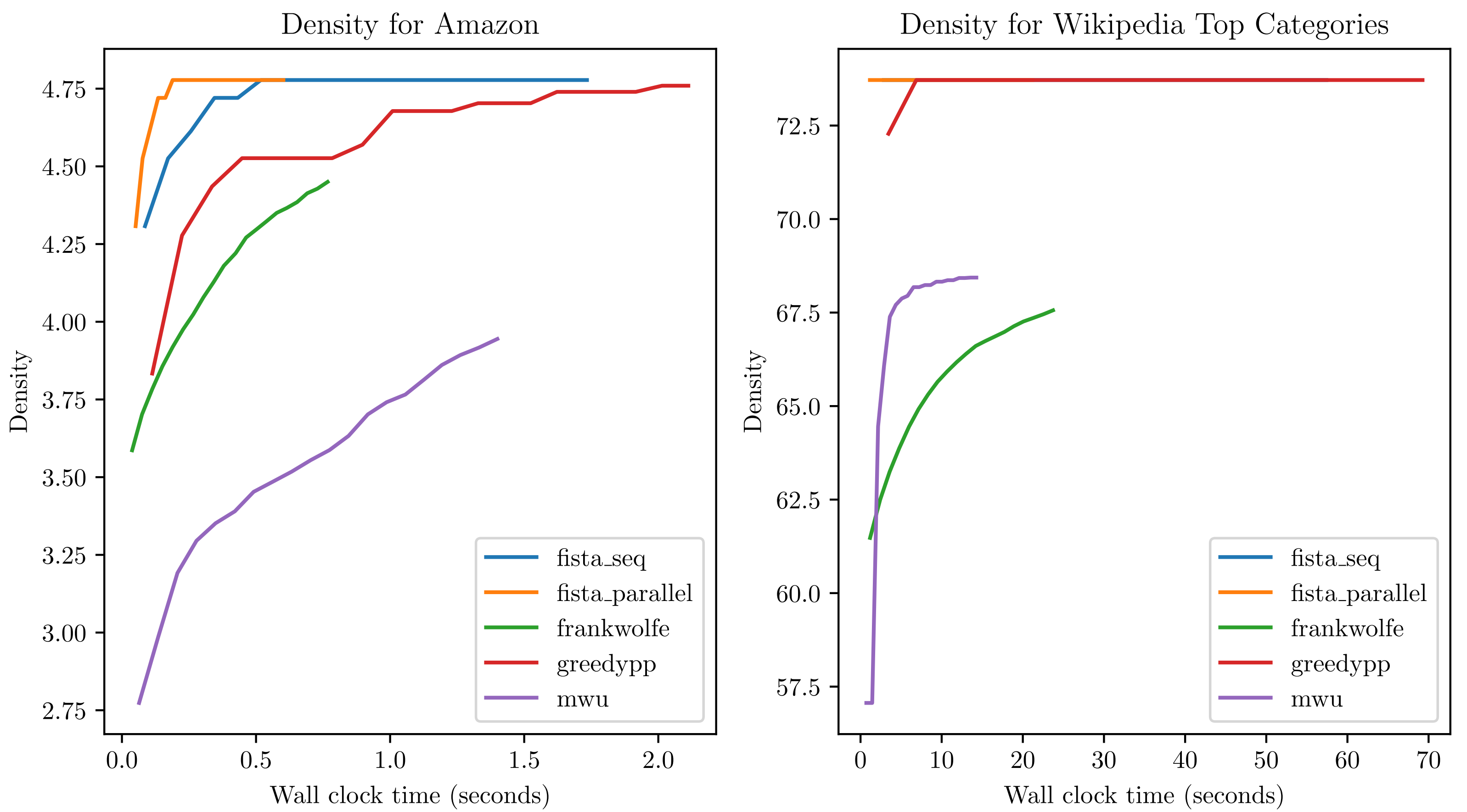
- SuperGreedy++ is **not** Frank-Wolfe on Fujisghige's QP
- So, what is it?
- **Main claim:** SuperGreedy++ is a *noisy* or *approximate* version of a *variant* of Frank-Wolfe
- Can generalize Frank-Wolfe convergence analysis to show that SuperGreedy++ also converges
- New proof has weaker convergence bound but gives additive guarantees. Also shows that SuperGreedy++ converges to the full dense decomposition vector rather than just max density

Iterative Algorithms for DSG and Empirical Evaluation

[HQC NeuRIPS'22]

- Focus on DSG and dense graph decomposition
- Algorithms
  1. Greedy++
  2. Frank-Wolfe on quadratic program starting with greedy solution as starting point [Danisch-Chan-Sozio'17]
  3. *Accelerated* proximal gradient method on quadratic program (FISTA). Main observation is that projection oracle is O(m) time so iterations are quite fast and parallelizable.
  4. MWU based algorithm
- Unlike Greedy++ other algorithms produce "fractional" solutions and need to be rounded. Introduce "fractional peeling" a heuristic with some theoretical support


Density for Roadnet CA
Density
1.4
1.5
1.6
1.7
1.8
1.9
0
2
4
6
8
10
Wall clock time (seconds)
fista_seq
fista_parallel
frankwolfe
greedypp
mwu
Density for Roadnet PA
Density
1.4
1.5
1.6
1.7
1.8
0
1
2
3
4
5
Wall clock time (seconds)
fista_seq
fista_parallel
frankwolfe
greedypp
mwu



Density for Amazon
Density
Wall clock time (seconds)
4.75
4.50
4.25
4.00
3.75
3.50
3.25
3.00
2.75
0.0
0.5
1.0
1.5
2.0
fista_seq
fista_parallel
frankwolfe
greedypp
mwu
Density for Wikipedia Top Categories
Density
Wall clock time (seconds)
72.5
70.0
67.5
65.0
62.5
60.0
57.5
0
10
20
30
40
50
60
70
fista_seq
fista_parallel
frankwolfe
greedypp
mwu

Iterative Algorithms for DSG and Empirical Evaluation

[HQC NeuRIPS'22]

- FISTA based algorithm seems to be the consistent winner but Greedy++, Frank-Wolfe also competitive. MWU quite slow

- Fractional peeling is very important for performance

See paper for detailed plots

Take aways

- SuperGreedy and SuperGreedy++: simple iterative algorithms for *any supermodular density function*
- For DSG, a new FISTA based algorithm that seems superior to other methods. Fractional peeling for rounding that applies for other methods as well
- Frank-Wolfe vs SuperGreedy++: former competitive but fractional while latter is "combinatorial"
- p-mean DSG is NP-Hard for $p < 1$. See [CT'23] for results and open problems

Open Problem

Tight analysis of Greedy++

- Recall conjecture is $O\left(\frac{1}{\epsilon^2}\right)$ iterations
- Worst example known to us: $\Omega\left(\frac{1}{\epsilon}\right)$ iterations for $(1-\epsilon)$ approximation
- Our bound: $O(\frac{1}{\epsilon^2}\ \frac{\Delta(G)}{\lambda^*}\ \log n)$

Thanks!