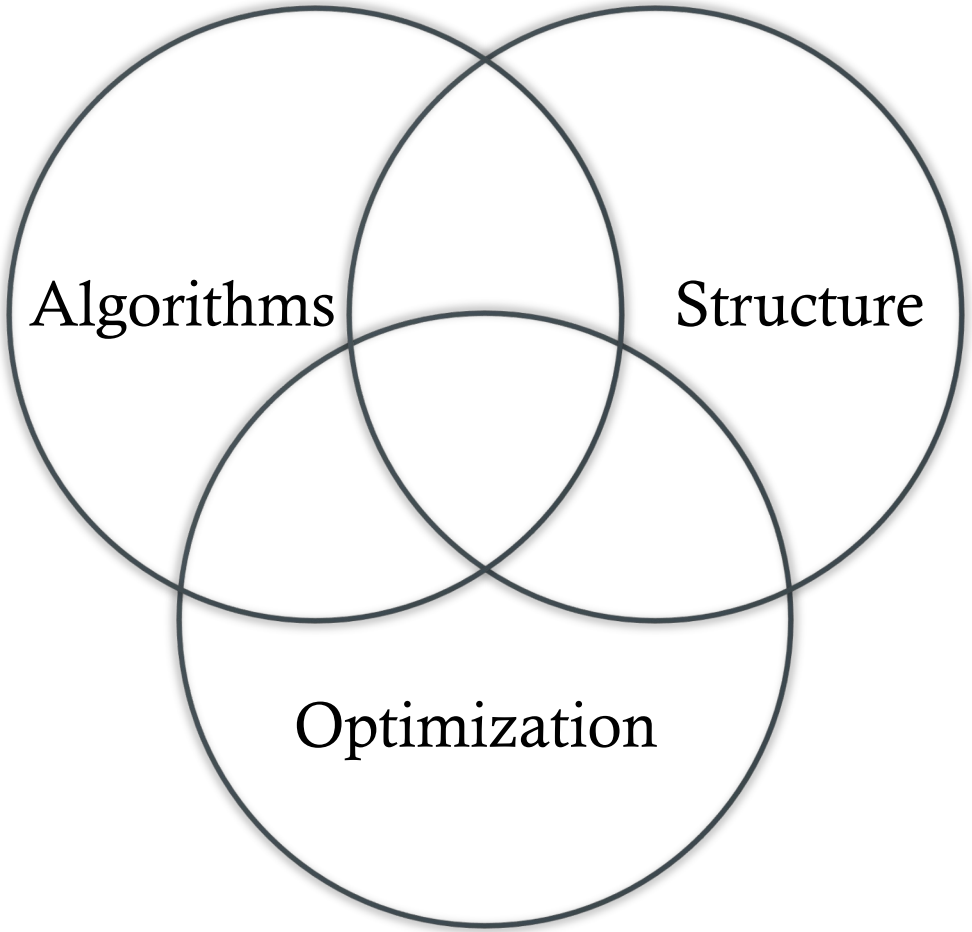
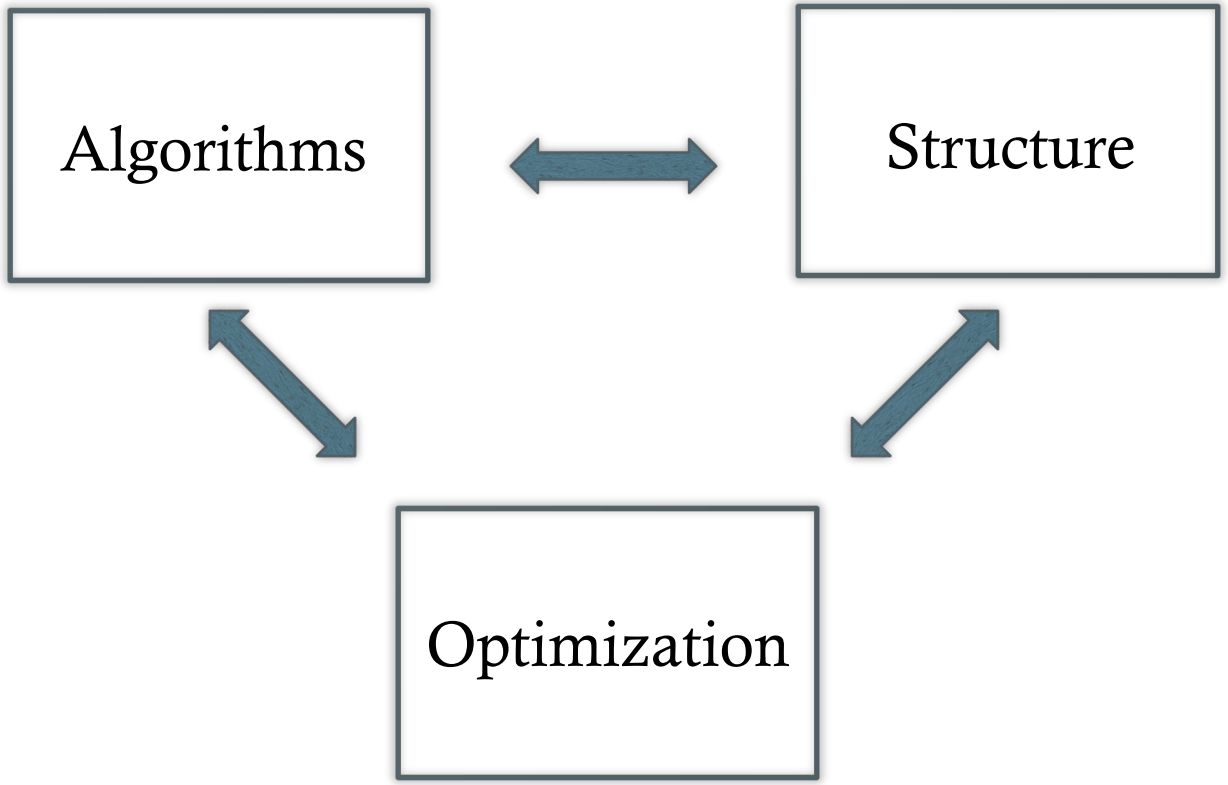
Densest Subgraph: A case study on the interplay between algorithms, structure, and optimization

Chandra Chekuri
Univ. of Illinois, Urbana-Champaign
(IISc Visiting Professor)

Based on joint works with
Harb ElFarouk (UIUC), Kent Quanrud (Purdue) and Manuel Torres (UIUC → JP Morgan)

IISc, July 12, 2024


Algorithms
Structure
Optimization


Algorithms
Structure
Optimization

Densest Subgraph (DSG)

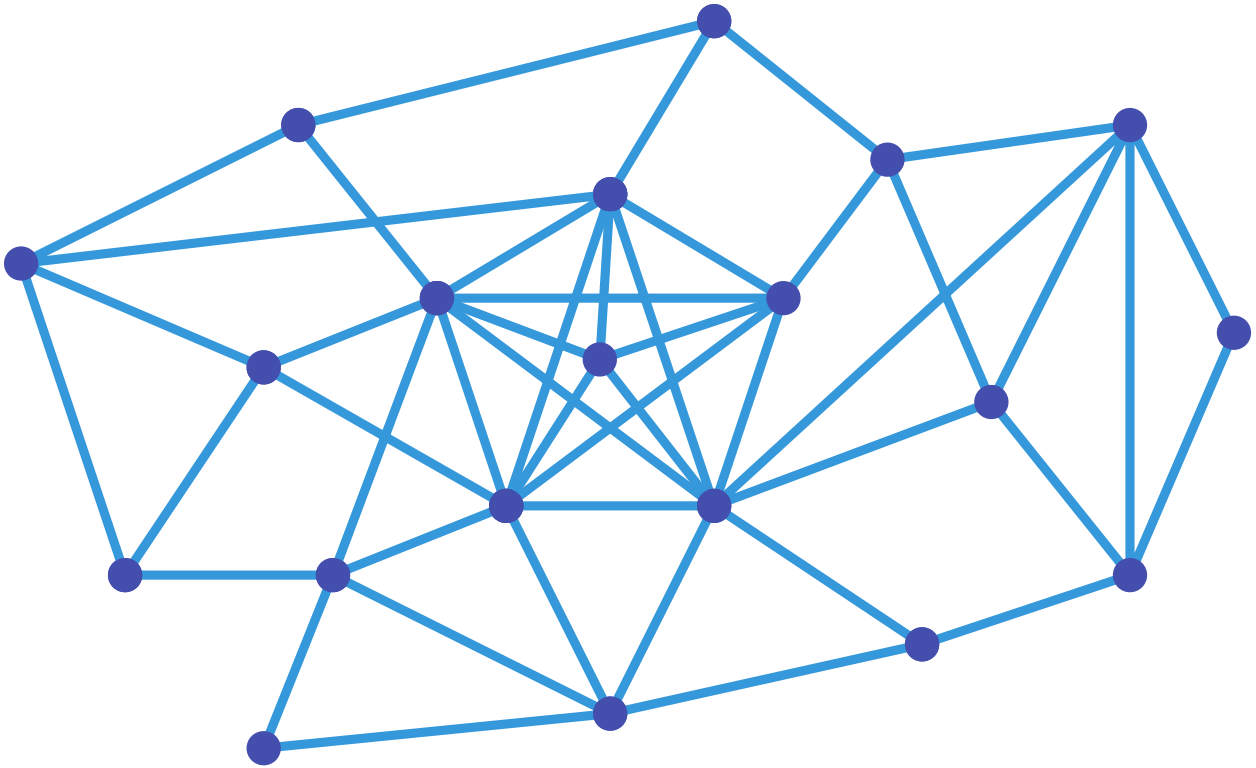
$G = (V, E)$ undirected graph

Find "dense" subgraph(s)

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

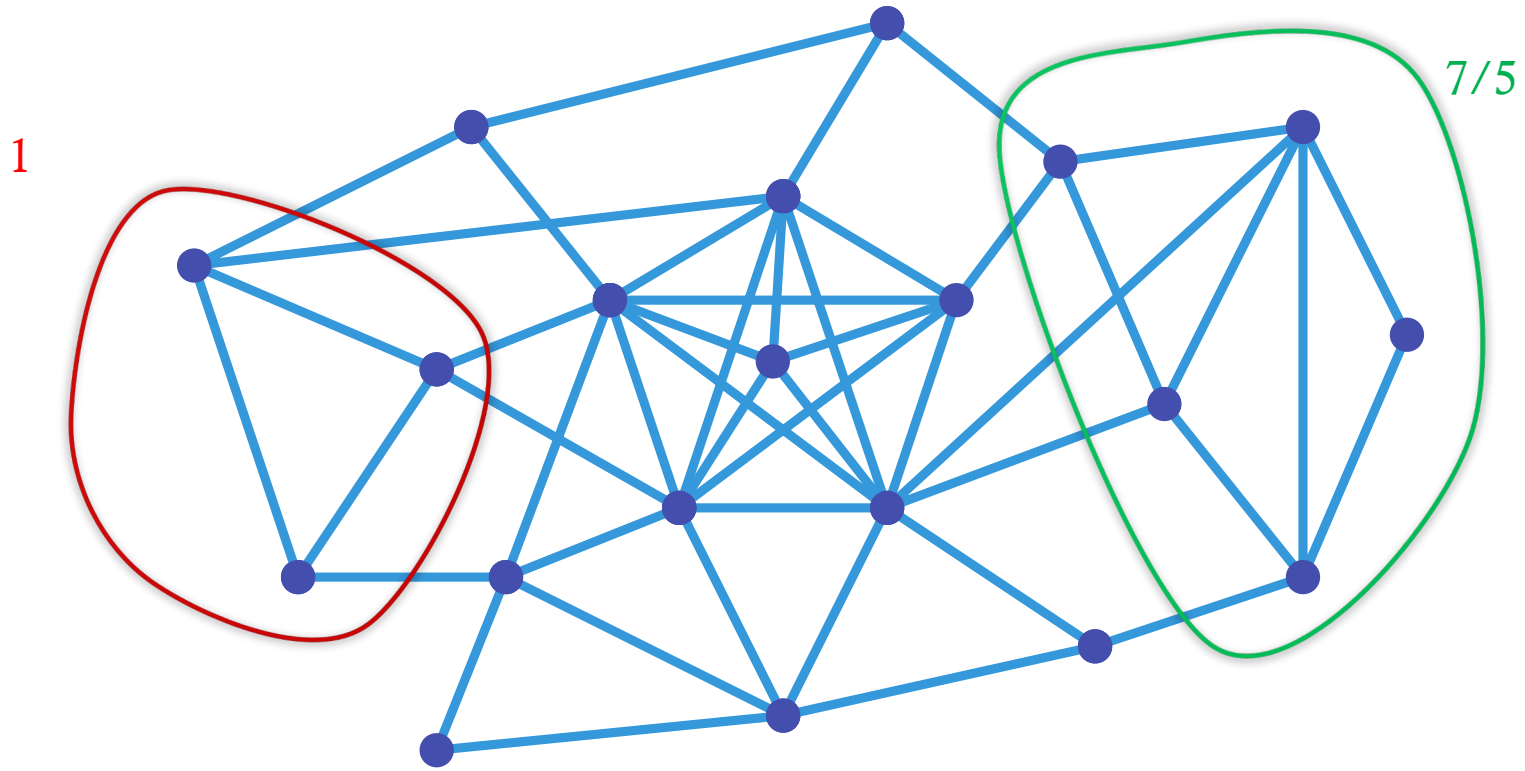
$$\lambda^* = \max_{S \subseteq V} \frac{|E(S)|}{|S|}$$

Example



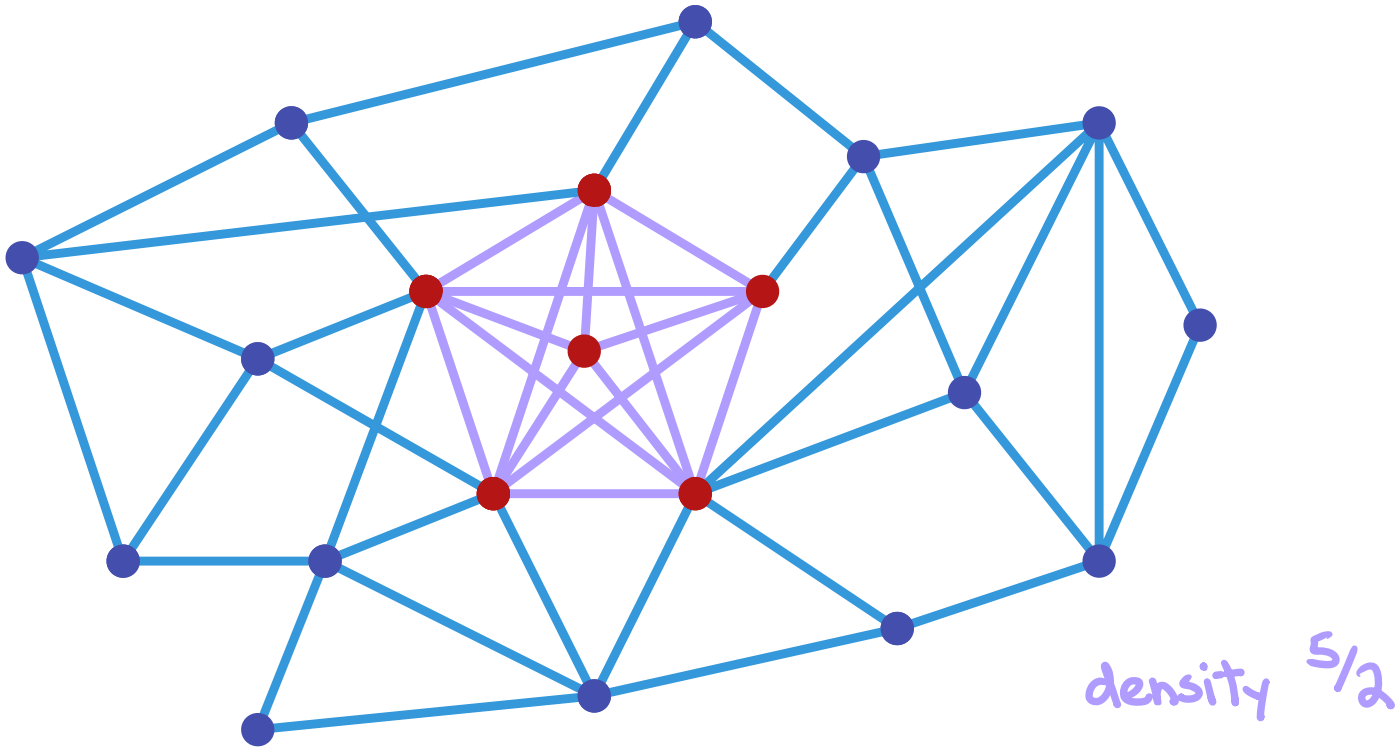
Figure/example from KQ

Example



Figure/example from KQ

Example



Dense Subgraph Discovery

- A basic problems in graph mining and network analysis --- many papers in data science conferences
  - Tutorial talk at SIAM Conf on Data Mining, 2021
  - Survey in ACM Computing Survey, 2024
- Several variants studied in practice and theory
  - Triangle density, k-clique density
  - Hypergraphs
  - p-mean density
  - Constrained versions
  - Directed graphs

Theoretical Connections

- Network Flow
- Arboricity of a graph, orientations etc
- Matroids and sub/supermodularity
- Densest k-subgraph and related problems
- Recent surprising application to Feedback Vertex Set
- …

Efficient Algorithms

DSG is polynomial-time solvable

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

Part 1: Algorithms

Need for new algorithms

- *very large* graphs in modern data analysis which requires simpler and faster algorithms
- variations and generalizations of DSG: different objective functions, hypergraphs, etc
- beyond sequential model: algorithms in streaming, parallel, dynamic settings
- …

Change in perspective: *approximation and heuristics*

[Charikar'00]

- Developed an exact LP relaxation for DSG
- Showed that a *simple linear-time* greedy algorithm yields a ½ - **approximation**

Algorithm's simplicity, efficiency, and practical performance made it a good choice for large data sets and was generalized to several other objectives

Greedy Peeling Algorithm

[Asahiro etal '00, Charikar '00]

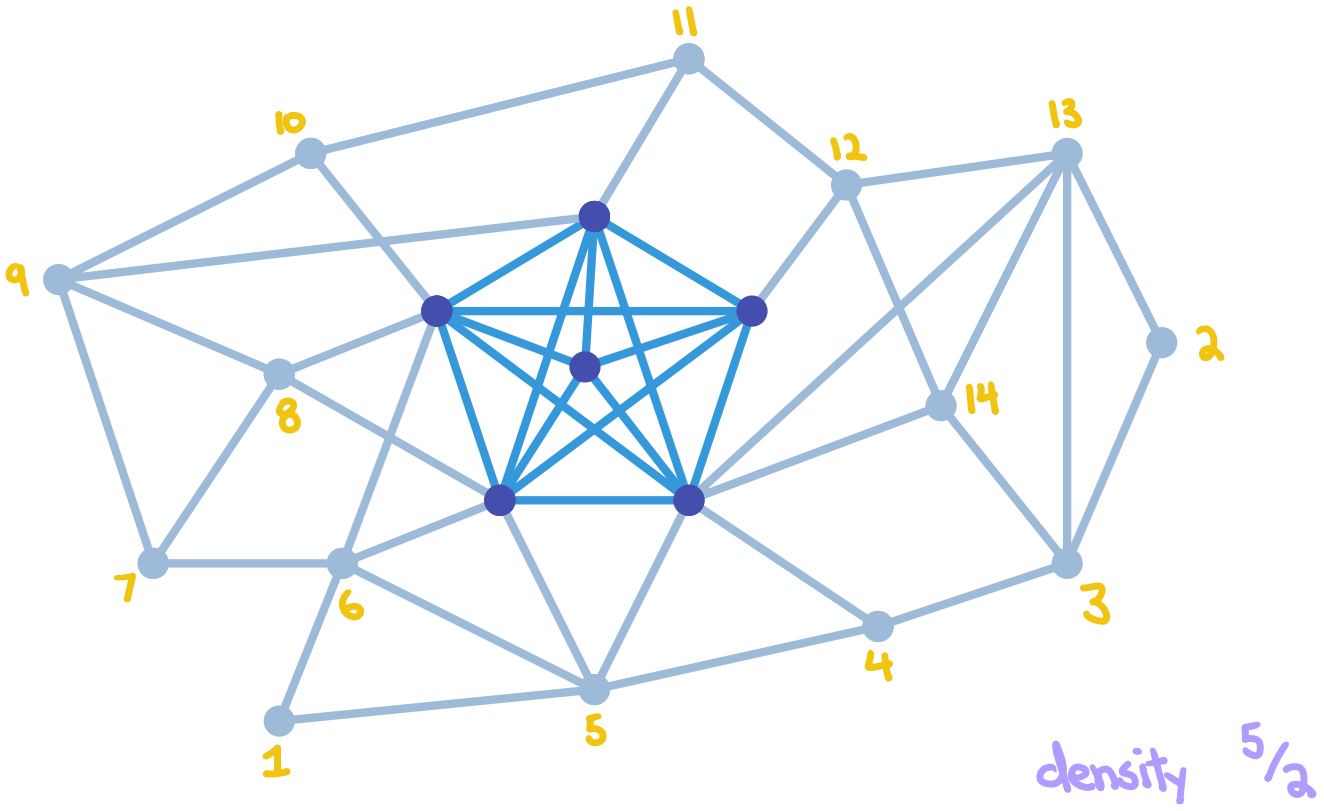
- For $i = 1\ to\ n$ do
  - v_i is in *min-degree* vertex in G
  - $G \leftarrow G - v_i$

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

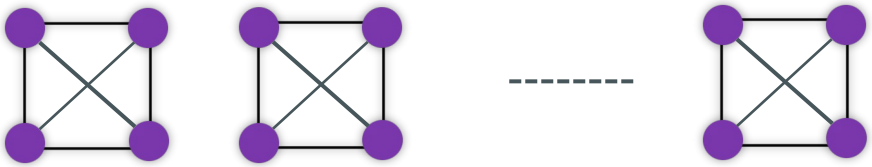
Known fact: Greedy gives *k-core decomposition of a graph*

Theorem: [Charikar'00] Greedy is a ½ approximation for DSG

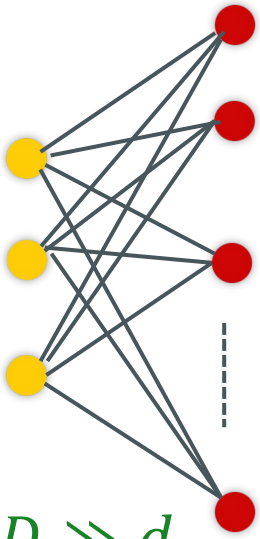
Example



Lower Bound Example



K_{d+1}

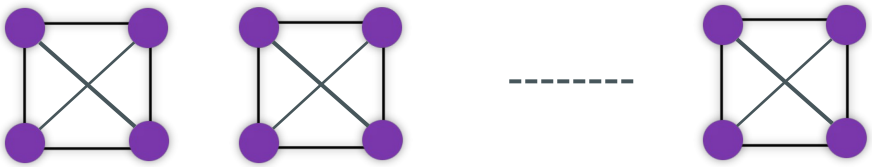


$K_{d,D}$ $D \gg d$

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

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

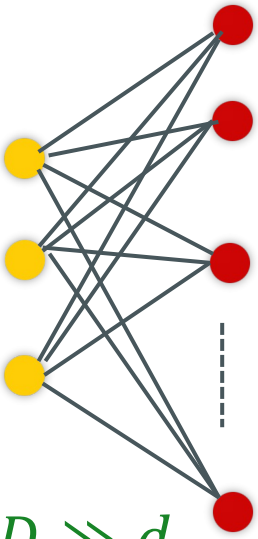
Lower Bound Example



K_{d+1}

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

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



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

Peeling order

Improved Heuristics

DSG is poly-time solvable

Question: is there a simple and fast algorithm that converges to an *optimal solution*? Avoid complexity of network flow

Several approaches to achieve a $(1-\epsilon)$ approximation via iterative methods starting with [Bahmani-Goel-Munagala'14]

Greedy++

Flowless: Extracting Densest Subgraphs Without Flow Computations

Digvijay Boob
Georgia Tech
digvijaybb40@gatech.edu

Yu Gao
Georgia Tech
ygao380@gatech.edu

Richard Peng
Georgia Tech
rpeng@cc.gatech.edu

Saurabh Sawlani
Georgia Tech
sawlani@gatech.edu

Charalampos E. Tsourakakis
Boston University
ctsourak@bu.edu

Di Wang*
Google AI
wadi@google.com

Junxing Wang
CMU
junxingw@cs.cmu.edu

October 18, 2019

Iterative Peeling

[BGPSTWW'20]

- Goal: improve ½ approx to $(1-\epsilon)$ approx.

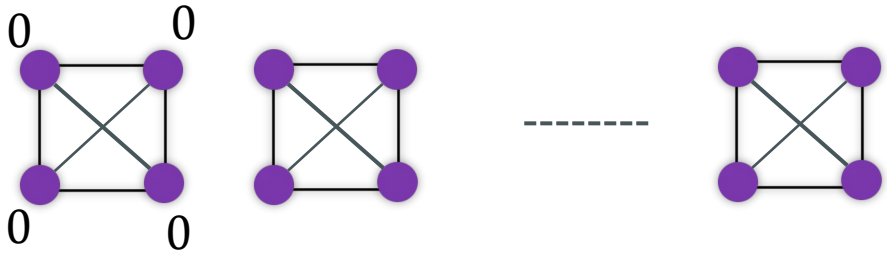
- Simple heuristic inspired by Dual-LP and MWU
- Peel several times by adjusting "load"
- Creates a new ordering in each iteration
- Pick best suffix among all orderings

Iterative Peeling

Greedy++

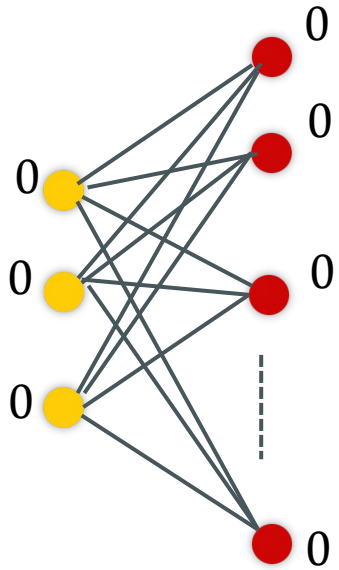
- $load(v, 0) = 0 \quad for\ all\ v$
- For $t = 1\ to\ T$ do
  - $G' \leftarrow G$
  - For $i = 1\ to\ n$ do
    - $v_{t,i} \leftarrow argmin_v \deg(v) + load(v, t-1)$
    - $load(v_{t,i}, t) = load(v_{t,i}, t-1) + \deg(v_{t,i})$
    - $G' \leftarrow G' - v_{i,t}$
- $S_{t,i} \leftarrow \{v_{t,i}, \dots,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

Example: start with 0 loads



K_{d+1}

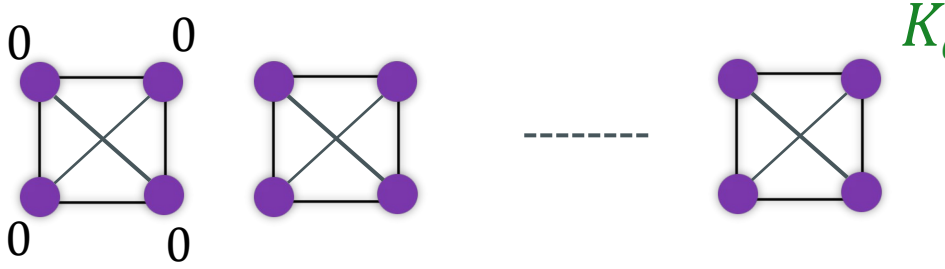
$\lambda^* \simeq d$



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

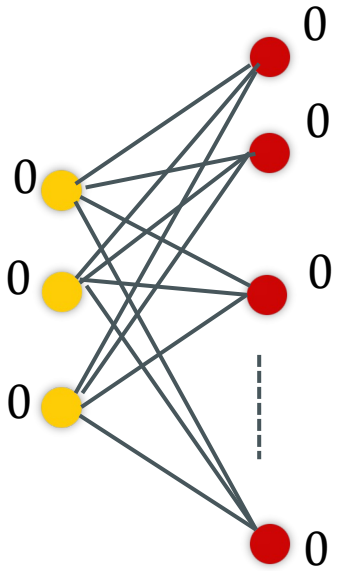
$K_{d,D}$ $D \gg d$

Peeling with load + degree



K_{d+1}

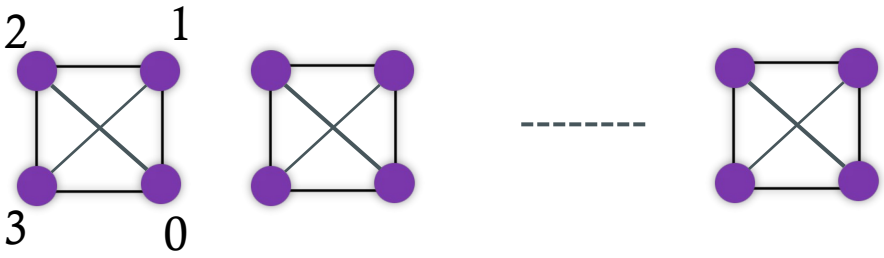
$\lambda^* \simeq d$



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

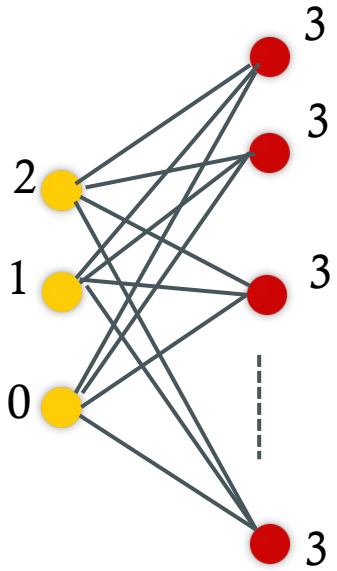
$K_{d,D} \quad D \gg d$

Update loads after peeling



K_{d+1}

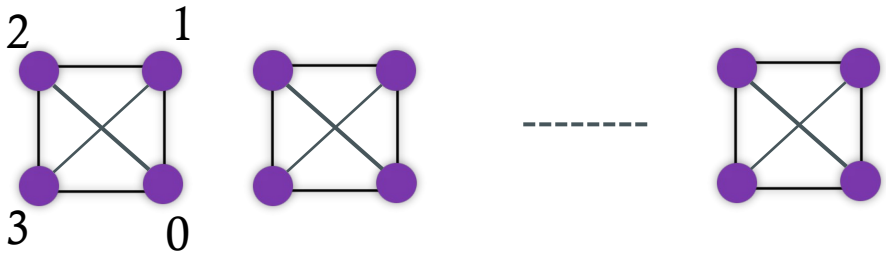
$\lambda^* \simeq d$



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

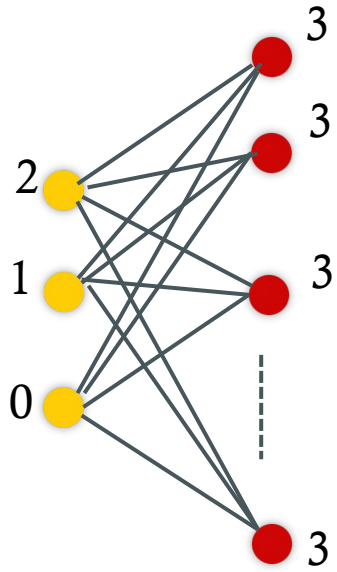
$K_{d,D} \quad D \gg d$

Peel with load + degree

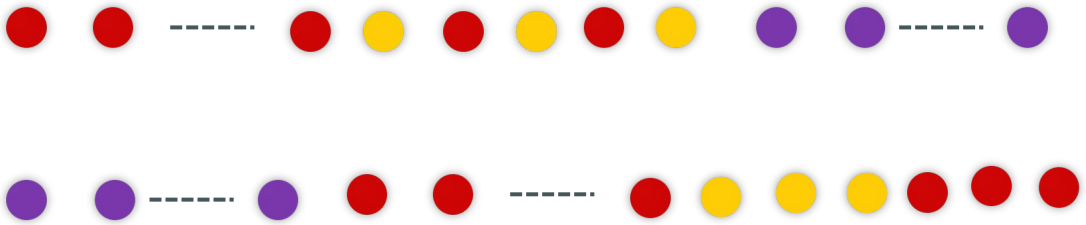


K_{d+1}

$\lambda^* \simeq d$

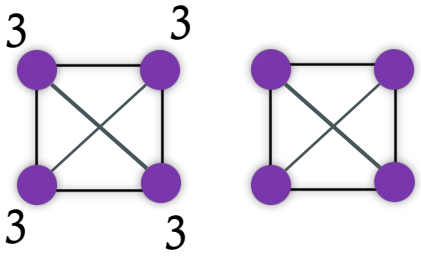


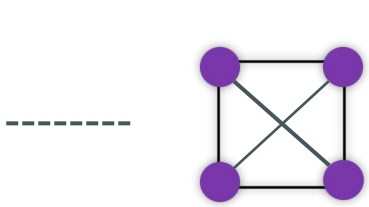
Peeling: $\lambda \simeq \frac{d}{2}$



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

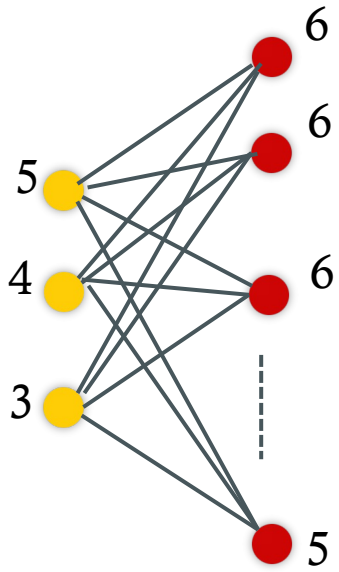
Update loads



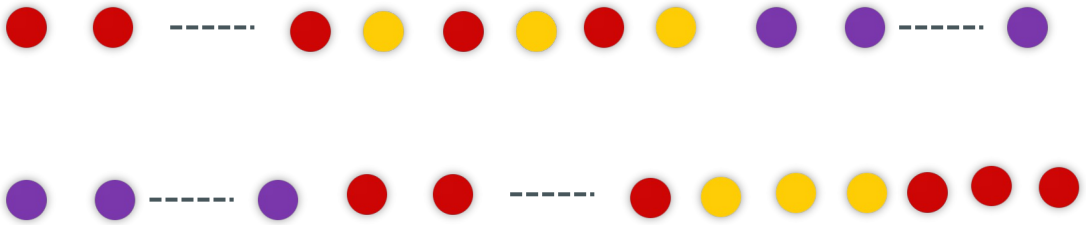


K_{d+1}

$\lambda^* \simeq d$



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



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

Performance and Conjecture

Algorithms works very well in practice.

Each iteration is extremely simple

Converges very quickly to optimal solution on many real-world graphs

[BGPSTWW'20]

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

Convergence

[C-Quanrud-Torres'22]

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

Open problem: close the gap between conjecture and above bound

Proof of Conjecture

Proof not so easy/obvious. Why?

Partial answer: problem/conjecture not really about graphs but applied in a more general structure

Part 2: Structure

Efficient Algorithms

DSG is polynomial-time solvable

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

An abstract problem formulation

DSG: given graph $G = (V, E)$, $\max_{S \subseteq V} \frac{|E(S)|}{|S|}$

Instead of a graph, a real valued set function $f: 2^V \rightarrow R_+$

$$\max_{S \subseteq V} \frac{f(S)}{|S|}$$

Question: which functions admit good algorithms?

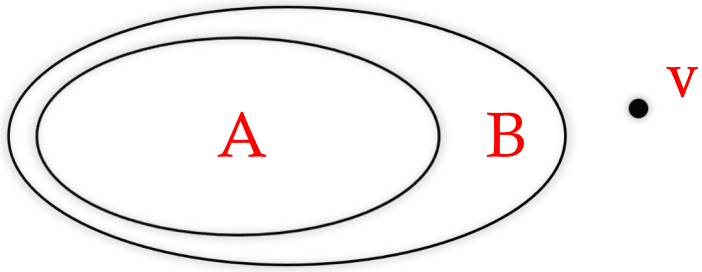
Sub and Supermodularity

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:

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



diminishing marginal utility property

Sub and Supermodularity

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

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

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

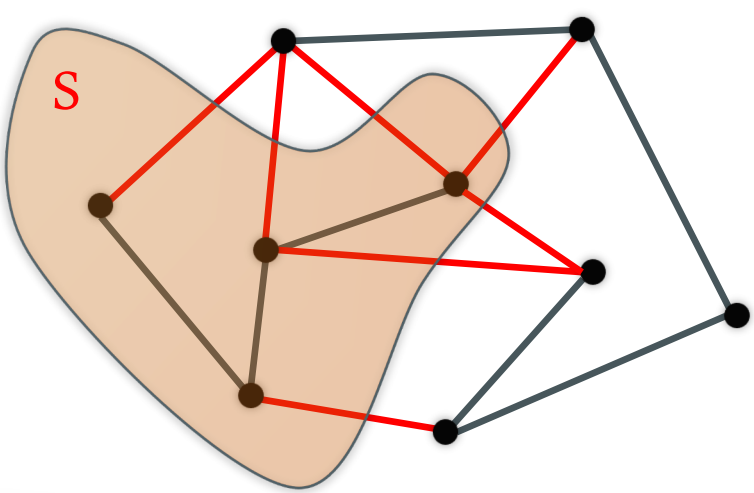
Supermodular:

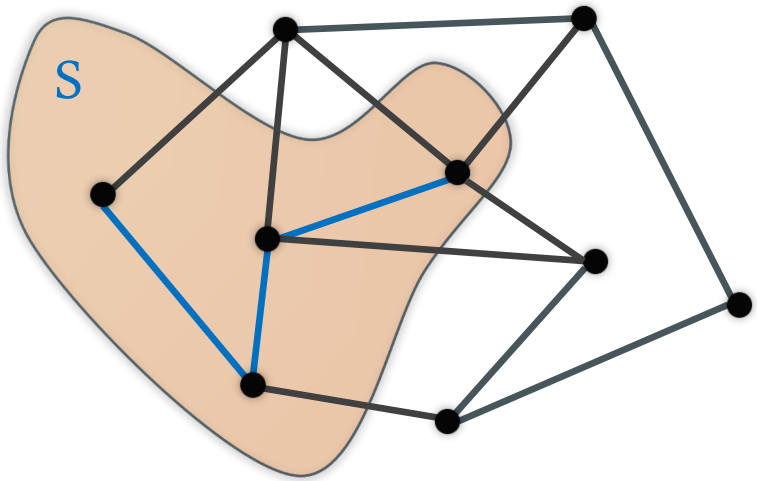
$$f(v \mid B) \geq f(v \mid A) \qquad A \subset B, v \in B - A$$

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

Poly-time via submodular function minimization, a classical result in combinatorial optimization.

However, the general algorithm is quite slow

Peeling and DSS

DSS: supermod function $f: 2^V \to R_+$ find $\max_S \frac{f(S)}{|S|}$

Questions:

Can we generalize Greedy and Greedy++ to DSS?

How do they perform?

Greedy Peeling for DSS

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

SuperGreedy

- 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 $argmax_i \frac{|f(S_i)|}{|S_i|}$

Iterative Peeling for DSS

Given supermodular $f: 2^V \rightarrow R_+$ 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 $argmax_{t,i} \frac{|f(S_{t,i})|}{|S_{t,i}|}$

Results for DSS

Greedy peeling is about supermodularity!

- SuperGreedy leads to max-core decomposition
- SuperGreedy is a $\frac{1}{c_f}$ approximation for DSS. Unifies all known results for special cases.
- SuperGreedy++ converges to a $(1-\epsilon)$ approximation in $O\left(\frac{1}{\epsilon^2}\frac{\max_v f(v|V-v)}{\lambda^*}\log n\right)$ iterations

Connections and Implications

- Proof of convergence crucially relies on properties that are only apparent when working with supermodular functions
- Shows that there is a simple heuristic that converges to optimum for many problems of interest in dense subgraph discovery
- Dense decomposition

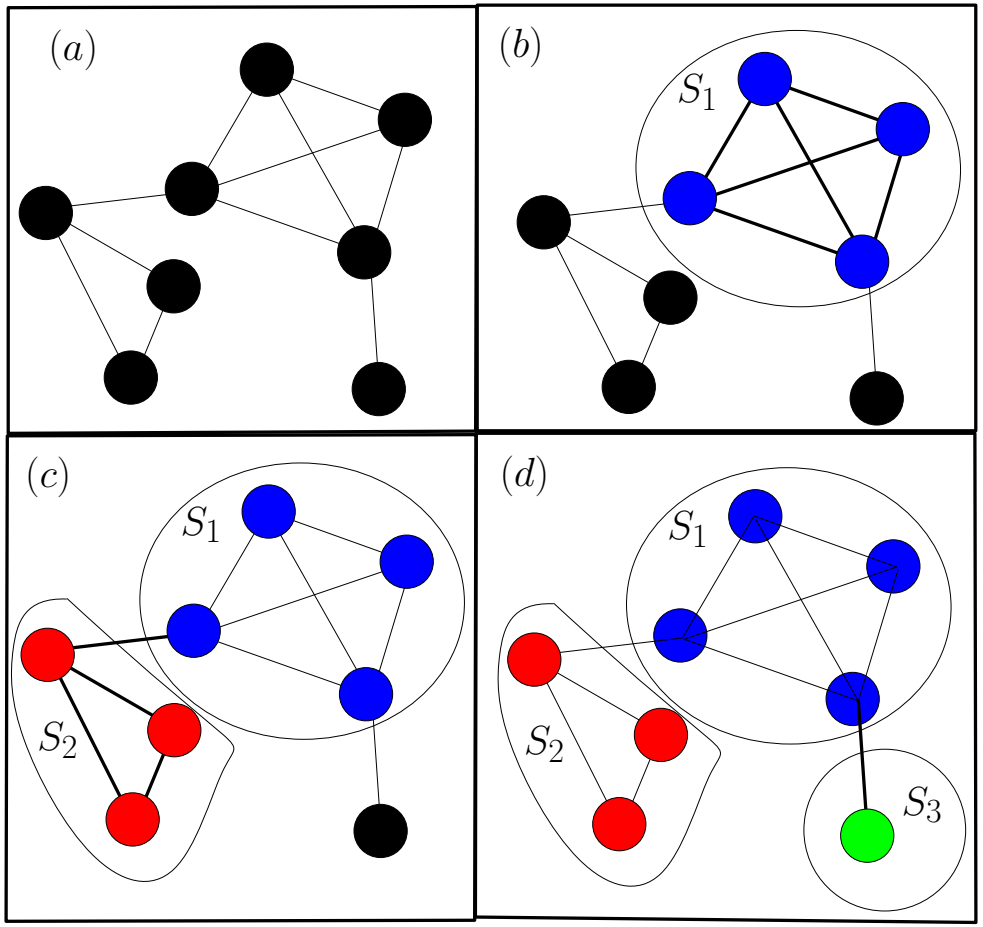
Dense Subgraph Decomposition

[Tatti-Gionis'15]

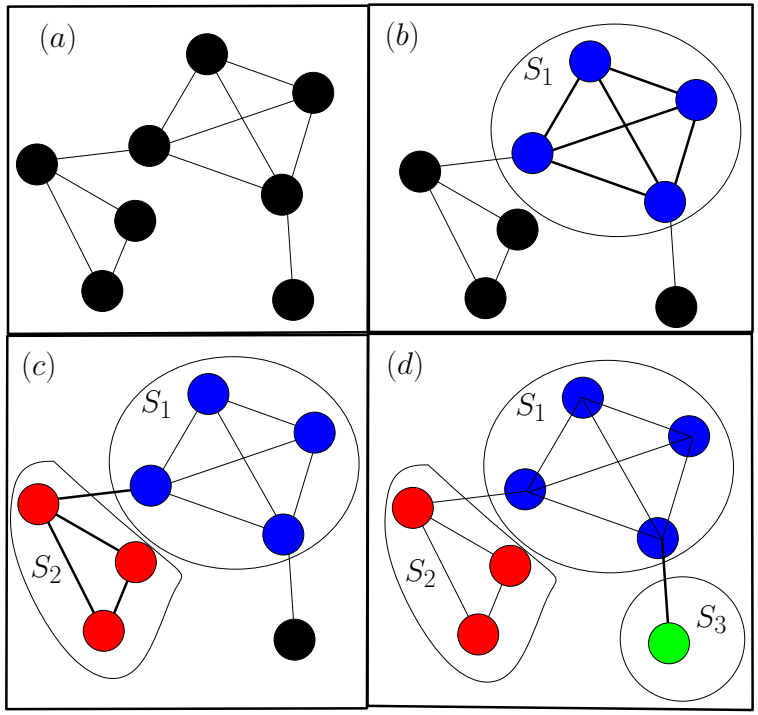
Every graph has a *unique maximal densest subgraph*

Every graph admits a *decomposition* or *partition* into decreasing density subgraphs

Dense Subgraph Decomposition



Dense Decomposition Vector



$$\begin{bmatrix} 6/4 \\ 6/4 \\ 6/4 \\ 6/4 \\ 4/3 \\ 4/3 \\ 4/3 \\ 1 \end{bmatrix}$$

For every v define den(v) as the density of the unique part that contains it

This defines a vector of densities which fully characterizes the dense decomposition

Dense Decomposition

[Harb-Quanrud-C'22]

Connection: Dense decomposition is a *supermodular* property. Several facts follow from known theory (due to [Fujishige'80])

[Harb-Quanrud-C'23]

Theorem: The loads in SuperGreedy++ converge to the dense decomposition vector!

Part 3: Optimization

Combining insights

- [Charikar'00] exact LP relaxation for DSG
- [Fujishige'80] a *quadratic program* that for any supermodular function yields the full dense decomposition vector

Fujishige's QP is implicit and is of exponential size

A simple quadratic program

[Harb-Quanrud-C'22]

For any graph G, a simple quadratic program that has the dense decomp vector as a *unique* optimum

$$\min \sum_{v \in V} \left(\sum_{uv} y_{uv} \right)^2$$

$$y(uv) + y(vu) = 1 \; for \; all \; uv \in E$$

$$y \geq 0$$

A simple quadratic program

[Harb-Quanrud-C'22]

Theorem: There is an O(m) time algorithm for *projecting* any point $z \in R^m$ onto the feasible region of the polytope.

That is, finding a point y in the feasible region that is closest to z in Euclidean distance

Implications

1. [HQC'22] A *new* simple and fast iterative algorithm based on accelerated gradient descent. Competitive with Greedy++ and faster in parallel settings
2. [HWC'23] Alternate proof of SuperGreedy++ convergence via connection to Frank-Wolfe method in convex optimization applied to Fujishige's QP. Proves that SuperGreedy++ converges to dense decomposition vector
3. [Nguyen-Ene ICML'24] A new algorithm based on random coordinate descent

Conclusions and take aways

- SuperGreedy and SuperGreedy++: simple iterative algorithms for *any supermodular density function.* First iteration already has a reasonable approximation.
- For DSG, new insights via convex optimization led to two new simple and fast converging algorithms besides Greedy++
- Several other results/connections. See refs

Open Problem

Tight analysis of Greedy++

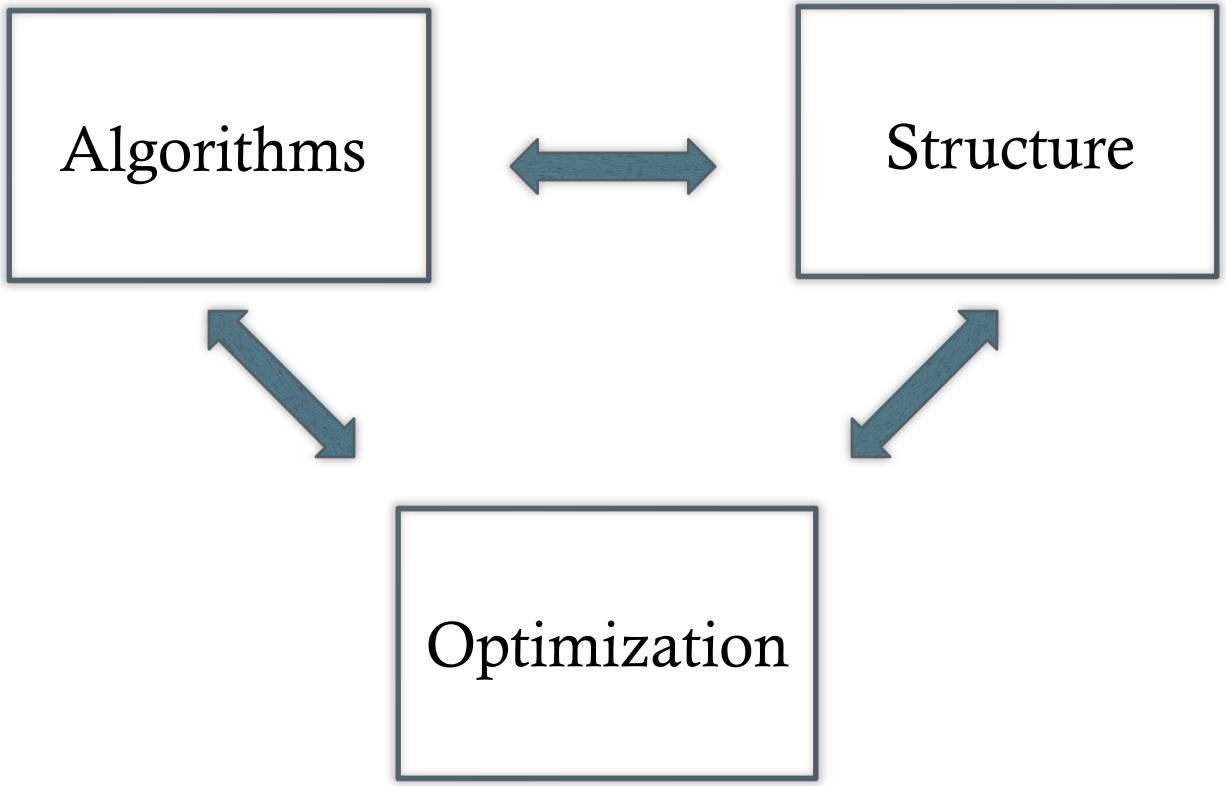
- Recall conjecture is $O\left(\frac{1}{\epsilon^2}\right)$ iterations
- Worst example known to us: $\Omega\left(\frac{1}{\epsilon}\right)$ iterations for $(1-\epsilon)$ approximation
- Our bound: $O\left(\frac{1}{\epsilon^2}\ \frac{\Delta(G)}{\lambda^*}\ \log n\right)$

What is the true worst-case convergence bound?

References

- "Densest Subgraph: Supermodularity, Iterative Peeling, and Flow", C. Chekuri, K. Quanrud, M. Torres. *Proc. of ACM-SIAM SODA, 2022.*
- "Faster and Scalable Algorithms for Densest Subgraph and Decomposition", E. Harb, K. Quanrud, C. Chekuri. *Proc. of NeuRIPS, 2022.*
- "Convergence to Lexicographically Optimal Base in a (Contra)Polymatroid and Applications to Densest Subgraph and Tree Packing", E. Harb, K. Quanrud, C. Chekuri. *Proc. of ESA (S), 2023.*
- "Multiplicative Weights Update, Area Convexity and Random Coordinate Descent for Densest Subgraph Problems", T. Nguyen, E. Ene. *Proc. of ICML, 2024. To appear.*

Thank You!



Charikar's LP Relaxation

Integer Programming Formulation

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

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

$$x, z \text{ binary variables}$$

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

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

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

Charikar's LP Relaxation

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

$$\sum_v z_v = 1$$

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

$$x, z \geq 0$$

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

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

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

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

Peeling and DSS

Question: What is approximation ratio for f?

Peeling and DSS

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

Supermodularity:

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

Peeling and DSS

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

Theorem/Observation: Greedy gives *core decomposition* for any supermodular function!

Iterative Peeling for DSS

[CQT'22]

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

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