Densest Subgraph: Supermodularity, Iterative Peeling and Flow

Chandra Chekuri
Univ. of Illinois, Urbana-Champaign

Based on joint work with Kent Quanrud and Manuel Torres

Northwestern, May 4, 2022

Densest Subgraph (DSG)

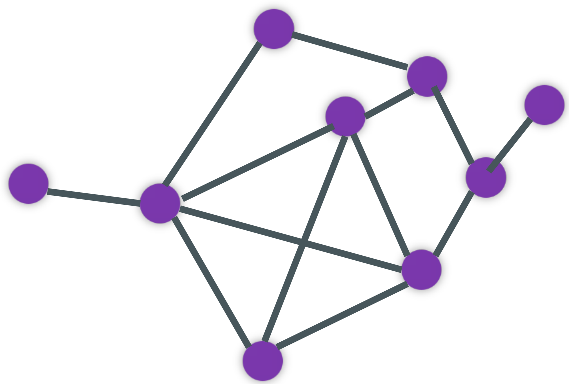
$G = (V, E)$ undirected graph

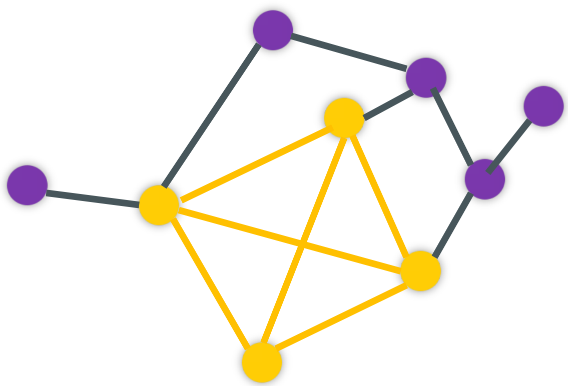
Find "dense" subgraph(s)

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

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

Example





$$\lambda^* = \frac{6}{4}$$

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

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

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

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

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
- …

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 …

Mainly about connections which are simple in retrospect

Rest of the talk

- Charikar's LP Relaxation
- Flow based approximation algorithm
- Peeling and Iterative Peeling
- Relating iterative peeling to LP solving via MWU

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

Solving LP Approximately

- Dual-LP is a *mixed packing and covering* LP
- Can obtain $(1-\epsilon)$ approx. in $O\left(m\frac{polylog(n)}{\epsilon^2}\right)$ time, even in parallel [Bahmani-Goel-Munagala'14]
- **Open question:** can we solve mixed packing and covering LPs in $O\left(N\frac{polylog(n)}{\epsilon}\right)$ time? Known for pure packing and covering [AllenZhu-Orecchia'14,Wang-Rao-Mahoney'15]
- $O\left(m\Delta\frac{polylog(n)}{\epsilon}\right)$ time for DSG [Boob-Sawlani-Wang'19]

Flow Reduction via Dual

Observed in [Boob et al]

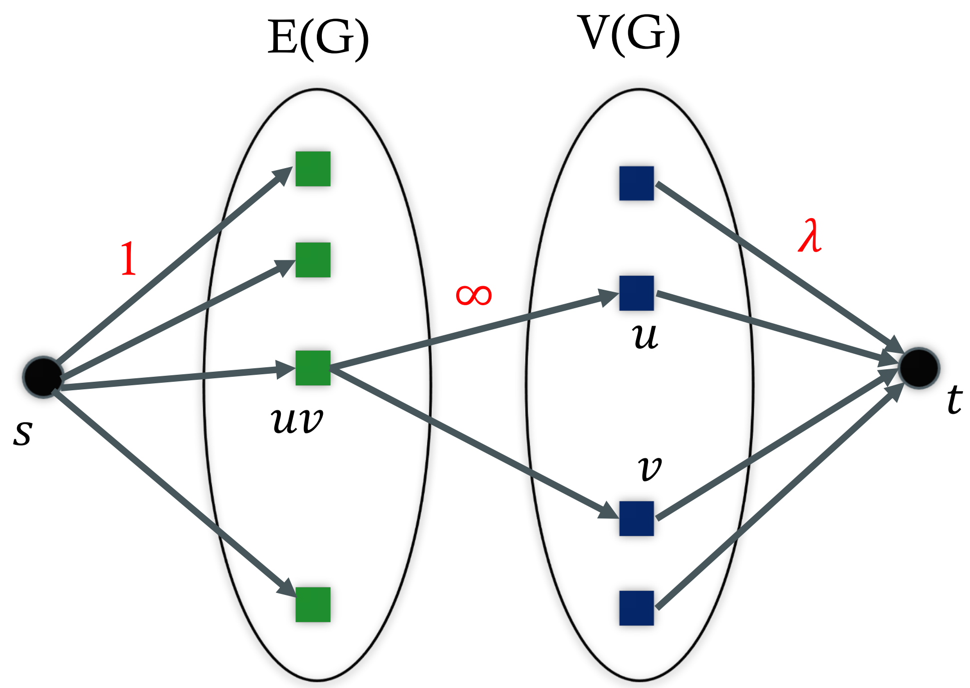
$\min D$

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

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

$y \geq 0$

Fractional perfect matching



Flow network H_λ

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

Flow based Approx Algorithm

Given value λ.

1. Construct H_λ
2. Run augmenting path algorithm: stop if shortest augmenting path length $\geq c \log n / \epsilon$

Theorem: If maxflow not reached then there exists subgraph in G with density $\geq (1 - \epsilon)\lambda$

Flow based Approx Algorithm

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

- Generalizes to hypergraphs
- Also yields faster approximation algorithm for densest *directed* subgraph via reduction

Peeling Algorithm

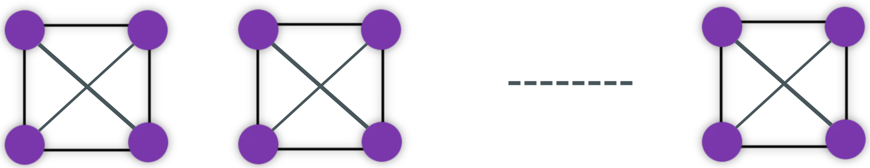
[Asahiro etal 00, Charikar 00]

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

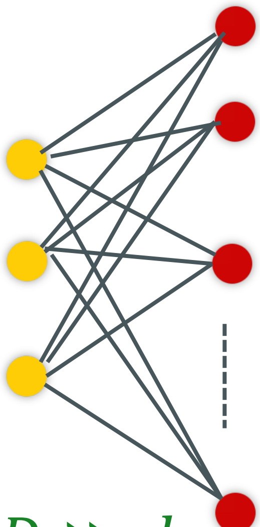
- $v_1, v_2, \ldots, v_n$ is ordering created by algorithm
- $S_i \leftarrow \{v_i, v_{i+1}, \ldots, v_n\}$
- Output $argmax_i \frac{|E(S_i)|}{|S_i|}$

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

(Tight) Example



K_{d+1}

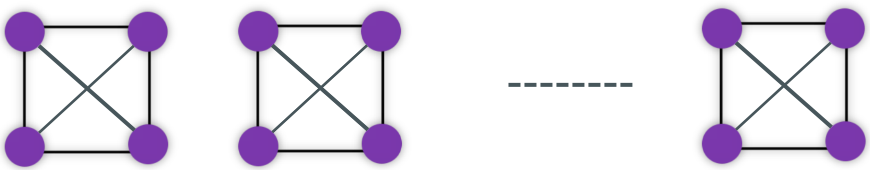


$K_{d,D}$ $D \gg d$

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

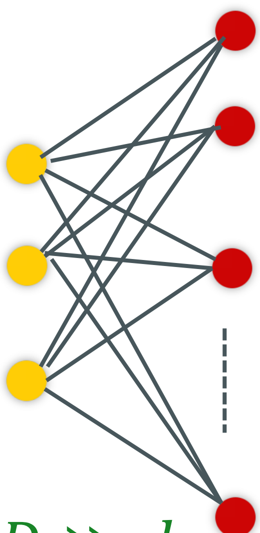
$\lambda(G) \simeq \frac{d}{2}$

(Tight) Example



K_{d+1}

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



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

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

Peeling order

Peeling and DSS

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

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

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

Peeling and DSS

Question: How can we characterize for general f?

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

Supermodularity: $\sum_{v \in S} f(v \,| S - v) \geq f(S) \Rightarrow c_f \geq 1$

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 of [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: $c_f = p + 1$

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

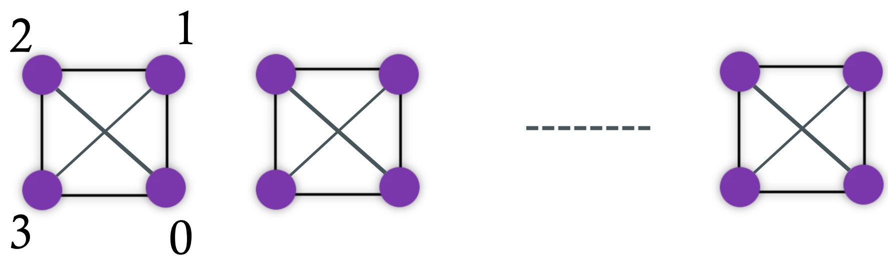
Iterative Peeling

[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 $argmax_{i,t} \frac{|E(S_{t,i})|}{|S_{t,i}|}$

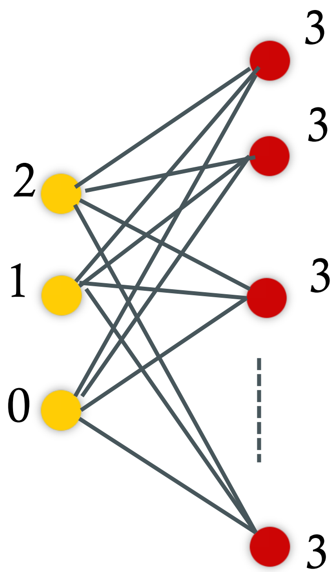
Example



K_{d+1}

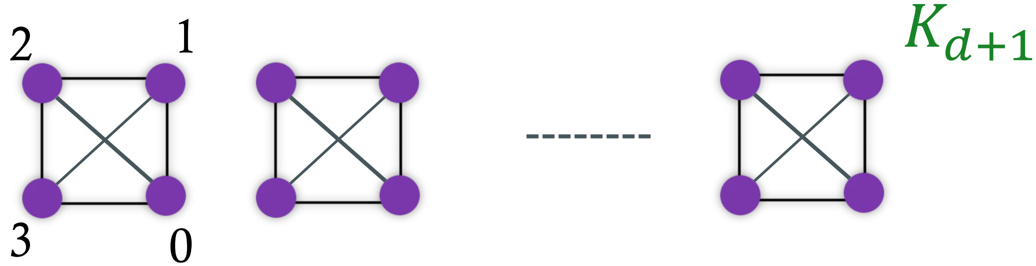
$\lambda^* \simeq d$

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



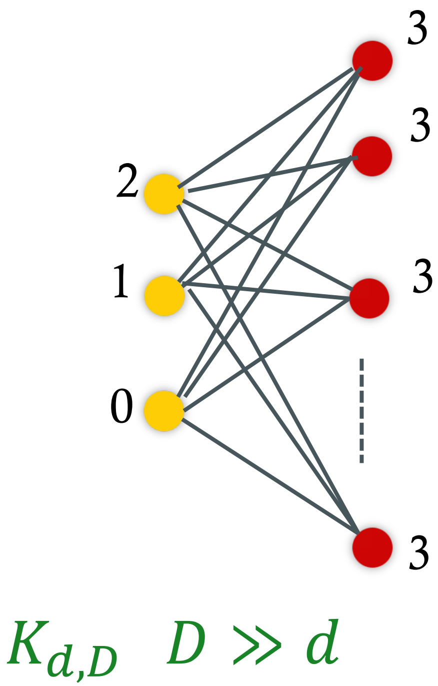
$K_{d,D}$ $D \gg d$

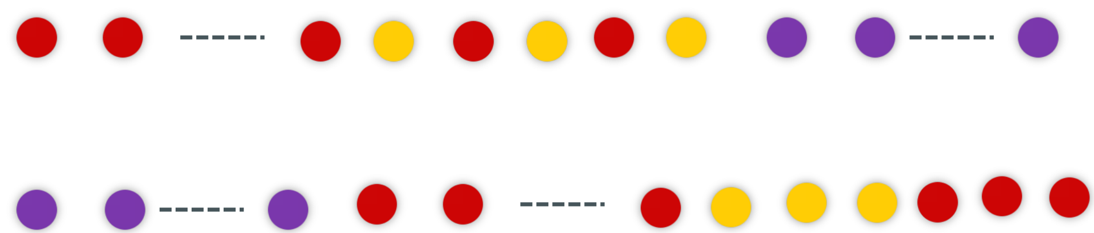
Example



$\lambda^* \simeq d$

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





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

Iterative Peeling for DSS

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

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

Proof Idea

- Express DSS as an LP relaxation
- Relate SuperGreedy++ iterations to a multiplicative-weight update (MWU) algorithm via LP

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

Charikar's LP

LP

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

$$\sum_v z_v = 1$$

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

$$x, z \geq 0$$

Concave Program

$$\max \sum_{uv \in E} \min(z_u, z_v)$$

$$\sum_v z_v = 1$$

$$z \geq 0$$

Lovasz Extension

$f: 2^V \to \mathbb{R}$ real valued set function

want to extend to continuous function $f: [0,1]^V \to \mathbb{R}$

Example: $V = \{v_1, v_2, v_3, v_4\}$ What is f(0.3, 0.7, 0, 0.1)?

Sort according to *decreasing* x values: v_2, v_1, v_4, v_3

$$f(0.3, 0.7, 0, 0.1)$$
$$= x_2\, f(v_2 \mid \emptyset) + x_1\, f(v_1 \mid \{v_2\}) + x_4 f(v_4 \mid \{v_2, v_1\}$$
$$+ x_3\, f(v_3 | \{v_2, v_1, v_4\})$$

Lovasz Extension

$f : 2^V \to \mathbb{R}$ real valued set function

A rounding interpretation:

$$\hat{f}(\mathrm{x}) = \mathbf{Ex}_{\theta \sim [0,1]}\left[\mathrm{f}\left(\mathrm{x}^{\theta}\right)\right]$$

where $\mathrm{x}^{\theta} = \{v \mid x_v \geq \theta\}$

Theorem:[Lovasz] $\hat{f}$ is convex iff f is submodular. $\hat{f}$ is concave iff f is supermodular.

Convex Relaxation for DSS

Supermodular func: $f: 2^V \to R_+$. Want $\max_S \frac{f(S)}{|S|}$

$$\max \hat{f}(z)$$
$$\sum_v z_v = 1$$
$$z \geq 0$$

Example: $G = (V, E), f(S) = |E(S)|$

$$\hat{f}(\mathrm{x}) = \sum_{uv \in E} \boldsymbol{min}(\boldsymbol{x}_u, \boldsymbol{x}_v)$$

Convex Relaxation for DSS

Supermodular func: $f: 2^V \rightarrow R_+$. Want $\max_S \frac{f(S)}{|S|}$

$$\max \hat{f}(z)$$

$$\sum_v z_v = 1$$

$$z \geq 0$$

Theorem: Relaxation is exact for DSS

Edmonds and Lovasz

Supermodular func: $f: 2^V \to R_+$

Consider all orderings/permutations of V

Given an ordering σ define a vector

$q(\sigma) \in R^V$ *where* $q_v(\sigma) = f(v \mid \{w \mid w \prec_\sigma v\})$

Example: $\sigma = v_2, v_4, v_3, v_1$

$q_{v_3}(\sigma) = f(v_2, v_4, v_3) - f(v_2, v_4)$

Edmonds and Lovasz

Supermodular func: $f: 2^V \to R_+$

Consider all orderings/permutations of V

Given an ordering σ define a vector

$q(\sigma) \in R^V$ *where* $q_v(\sigma) = f(v \mid \{w \mid w \prec_\sigma v\})$

Fact: $\hat{f}(\mathrm{x}) = \min_\sigma x^T q(\sigma)$.

Given x, the optimum ordering σ_x is to sort coordinates of x in *decreasing* order of x_v.

Rewriting Relaxations

$$\max \hat{f}(z)$$

$$\sum_v z_v = 1$$
$$z \geq 0$$

$$\min \sum_v z_v$$

$$\hat{f}(z) \geq 1$$
$$z \geq 0$$

OPT val = λ^*

OPT val = $1/\lambda^*$

Rewriting Relaxations

$$\min \sum_v z_v$$

$$\hat{f}(z) \geq 1$$

$$z \geq 0$$

OPT val = $1/\lambda^*$

$$\min \sum_v z_v$$

$$z^T q(\sigma) \geq 1 \quad for\ all\ \sigma$$

$$z \geq 0$$

OPT val = $1/\lambda^*$

Exponential sized ***LP***

Rewriting Relaxations

$$\min \sum_v z_v$$

$$z^T q(\sigma) \geq 1 \quad for\ all\ \sigma$$

$$z \geq 0$$

OPT val = $1/\lambda^*$

Exponential sized **LP**

$$\max \sum_\sigma y_\sigma$$

$$\sum_\sigma q_v(\sigma)\, y_\sigma \leq 1 \quad for\ all\ v \in V$$

$$y \geq 0$$

Dual **LP**

Ordering LP Relaxation

$$\max \sum_{\sigma} y_{\sigma}$$

$$\sum_{\sigma} q_v(\sigma)\, y_{\sigma} \leq 1 \quad for\ all\ v \in V$$

$$y \geq 0$$

- Packing LP
- Exponential # of variables but only n non-trivial constraints
- Amenable to MWU techniques

Solving Ordering LP via Multiplicative Weight Updates

- MWU: iterative algorithm for solving LPs
- Maintain (exponential) weights on constraints (dual variables)
- In each iteration solve a Lagrangean relaxation and take a small step along solution

$f: 2^V \to R$ is supermodular

For ordering σ of V, $q(\sigma)$ is a vector where

$$q_v(\sigma) = f(v \mid \{u \mid u \prec_\sigma v\})$$

$$max \sum_\sigma y_\sigma$$

$$\sum_\sigma q_v(\sigma)\, y_\sigma \leq 1 \quad for\ all\ v \in V$$

$$y \geq 0$$

1. $y^0 = \mathbf{0}$
2. $\mathrm{load}^0(v) = 1$ for all v
3. $\eta = \frac{1}{\epsilon} \log n$
4. For $t = 1$ to T do
   - $\sigma_t = argmin_\sigma \langle load^{t-1}, q(\sigma) \rangle$
   - $y^t = y^{t-1} + \frac{1}{\lambda^* T} \mathbf{1}_{\boldsymbol{\sigma_t}}$
   - For each v set $load^t(v) \leftarrow \exp(\eta \sum_\sigma y_\sigma^t q_v(\sigma))$
5. Output $y^T = \frac{1}{\lambda^* T} \sum_t \mathbf{1}_{\boldsymbol{\sigma_t}}$

$f: 2^V \to R$ is supermodular

For ordering σ of V, $q(\sigma)$ is a vector where

$$q_v(\sigma) = f(v \mid \{u \mid u \prec_\sigma v\})$$

$$\max \sum_v y_\sigma$$

$$\sum_\sigma q_v(\sigma)\, y_\sigma \le 1 \quad for\ all\ v \in V$$

$$y \ge 0$$

1. $y^0 = \mathbf{0}$
2. $load^0(v) = 1$ for all v
3. $\eta = \frac{1}{\epsilon} \log n$
4. For t = 1 to T do
   - $\sigma_t = argmax_\sigma \langle load^{t-1}, q(\sigma) \rangle$
   - $y^t = y^{t-1} + \frac{1}{\lambda^* T} \mathbf{1}_{\sigma_t}$
   - For each v set $load^t(v) \leftarrow \exp(\eta \sum_\sigma y_\sigma^t q_v(\sigma))$
5. Output $y^T = \frac{1}{\lambda^* T} \sum_t \mathbf{1}_{\sigma_t}$

MWU Analysis: Algorithm outputs $(1-\epsilon)$ approx if $T = \Omega(\frac{\Delta}{\epsilon^2 \lambda^*} \log n)$

Iterative Peeling and MWU

- MWU algorithm with LP naturally works with orderings of V which we see in SuperGreedy++
- SuperGreedy++ is *not* implementing standard MWU algorithm
- Why?
  - For graphs, given load(v) for each v
    - Output ordering according to decreasing order of loads
    - Static and does not add deg(v) correction term
    - Hence in first iteration *any* ordering is ok for MWU

Iterative Peeling and MWU

- SuperGreedy++ is *not* implementing standard MWU algorithm

- **Technical Lemma:** For appropriate parameter setting, each iteration of SuperGreedy++ yields a $(1+\epsilon)$ approximate ordering in MWU algorithm
- **Intuition:** deg is static while loads are increasing so initial Greedy step washes out eventually. Advantage of initial Greedy is its performance even after one iteration

Iterative Peeling and MWU

- SuperGreedy++ is *not* implementing standard MWU algorithm

- **Technical Lemma:** For appropriate parameter setting each iteration of SuperGreedy++ yields a $(1+\epsilon)$ approximate ordering in MWU algorithm
- MWU analysis is robust to approximate oracle
- Putting together yields convergence analysis

Summary

- **Fast approximate algorithm:** $(1-\epsilon)$ approximation for densest subgraph in $O\left(m\frac{polylog(n)}{\epsilon}\right)$ time. *Short augmenting paths suffice for density calculation*
- **SuperGreedy++:** simple iterative algorithm that converges for *any supermodular function*
- Other results in paper showcasing utility of supermodular perspective

Open Problems

- Tight analysis of iterative peeling
  - Worst example known to us: $\Omega\left(\frac{1}{\epsilon}\right)$ iterations for $(1-\epsilon)$ approximation
  - Is dependence on Δ, n, λ^* necessary? What about for DSS?
- Improved sequential, dynamic and parallel algorithms for DSG, DSS, and variants

Thanks!