MAKING CENTRALIZED (GRAPH) COMPUTATION FASTER, DISTRIBUTED AND (AT TIMES) BETTER

Devavrat Shah

Laboratory for Information Decision Systems

Massachusetts Institute of Technology

Joint work with

Vincent Blondel

UCL

Kyomin Jung

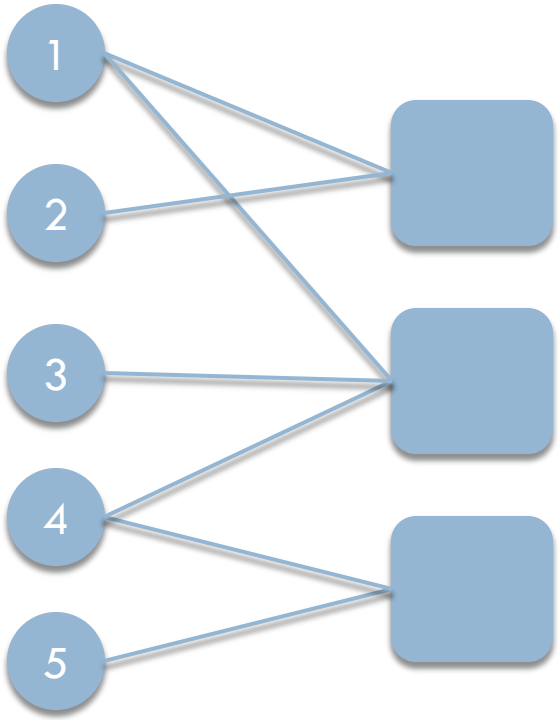
KAIST

Graphical models

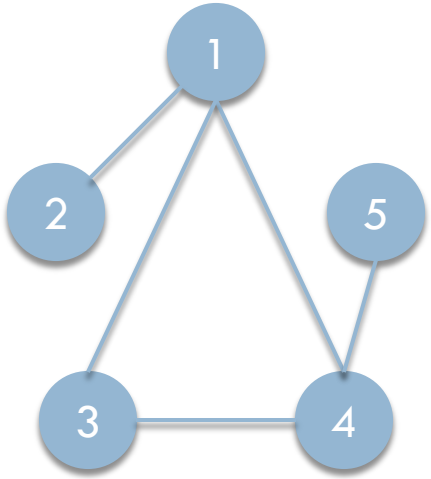
- Graphs provide canonical representation
  - To capture "interaction" of all sorts

- Examples
  - Social communications:
    - Who (e-)talks to whom
  - Cascading dynamics:
    - Failure in e-grid
  - Mutual (in)dependence:
    - Price of oil and political (in)stability

Example

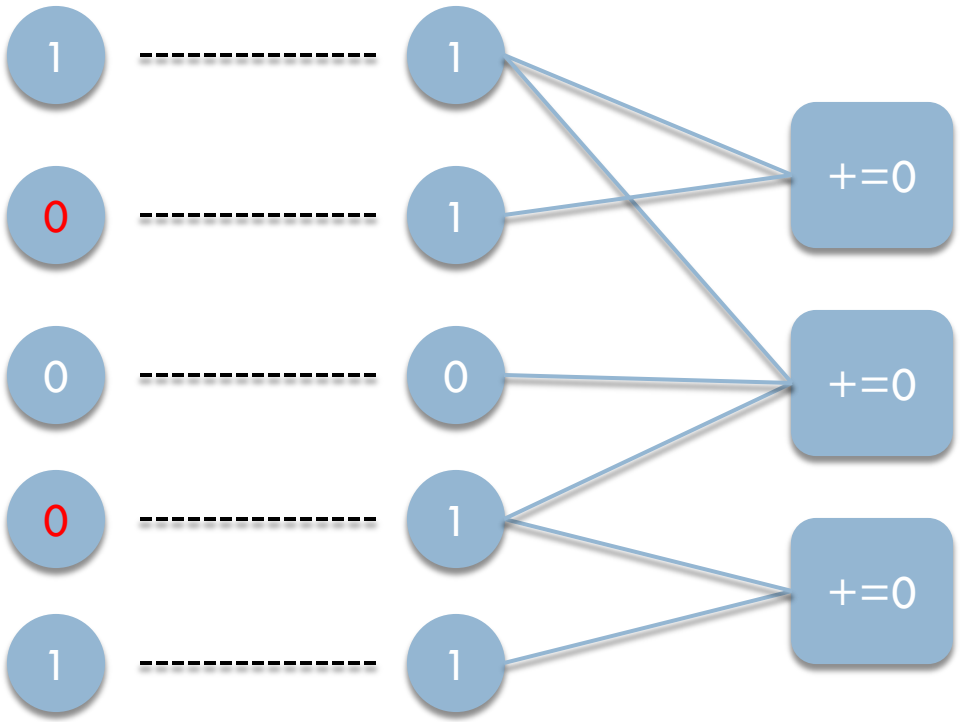


Ground Truth



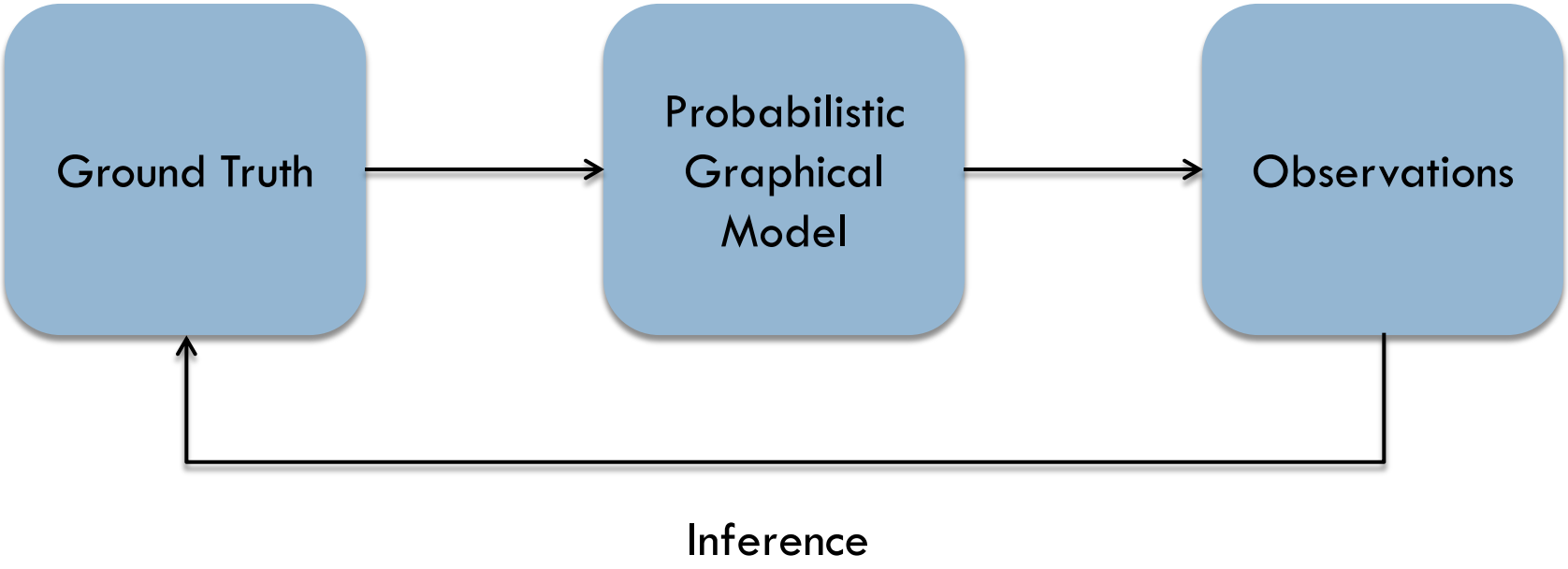
Observations

Example



Observations Ground Truth

Probabilistic model



- Maximum A Posteriori/Maximum likelihood estimation
  - A graph optimization problem

Inference: Clustering

- Newmann '06
  - Connections between nodes inside a cluster are less *likely to resemble randomly formed connections*

    (and likely to be more than random)

- Modularity metric:
  - Graph G=(V,E) with V={1,…,n}
    - d_i be degree of node i,
    - A_{ij} is 1 if (i,j) is an edge in E
  - Modularity of a subset C of V is

$$\mathbf{M}(C) = \sum_{ij \in C} \left(A_{ij} - \frac{d_i d_j}{2|E|} \right)$$

Inference: Clustering

- Newmann '06
  - Connections between nodes inside a clusters are less *likely to resemble randomly formed connections*

- Modularity metric:
  - Graph G=(V,E) with V={1,…,n}
    - d_i be degree of node i, A_{ij} is 1 if (i,j) is an edge in E
  - *Most likely* clustering optimizes overall modularity: choose clusters $C_1,\ldots,C_k$ so that they

$$\text{maximize} \sum_l \mathbf{M}(C_l) \text{ over } \textit{all possible choices}$$

Inference: Clustering

- Newmann '06
  - Connections between nodes inside a clusters are less *likely to resemble randomly formed connections*

- Modularity metric:
  - Graph G=(V,E) with V={1,…,n}
    - d_i be degree of node i, A_{ij} is 1 if (i,j) is an edge in E
  - *Most likely* clustering optimizes overall modularity:

$$\text{maximize} \sum_{ij \in V} \mathrm{I}(\chi(i) = \chi(j)) \left(A_{ij} - \frac{\mathrm{d_i d_j}}{2|\mathrm{E}|} \right) \text{ over } \chi : \mathrm{V} \rightarrow \{1,...,\mathrm{n}\}$$

    - *Computationally hard (to approximate)*

Inference: pair-wise graphical model

- Formally
  - Graph G = (V, E)
    - V = {1,…,N} correspond to N variables, say $X_1,\ldots,X_N$
      - Each variable takes value in finite alphabets, say {0,1}
    - E is set edges capturing "dependencies"
  - Joint distribution:

$$P(X_1 = x_1,\ldots, X_N = x_N) \propto \exp\left(\sum_{i\in V} \phi_i(x_i) + \sum_{(i,j)\in E} \psi_{ij}(x_i, x_j)\right)$$

    - Node potentials: ϕ_i for all $i \in V$
    - Edge potentials: ψ_{ij} for all $(i,j) \in E$

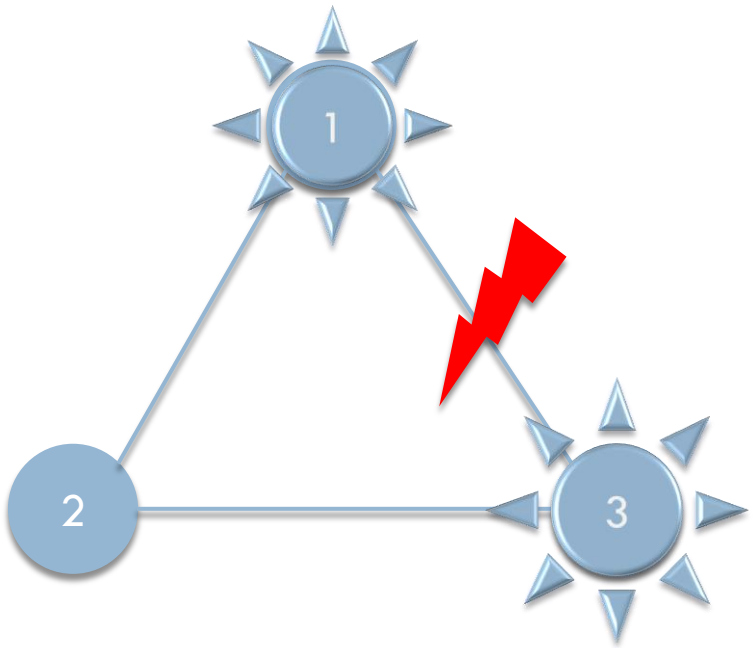
Inference: pair-wise graphical model

- Formally
  - Graph G = (V, E)
    - V = {1,…,N} correspond to N variables, say $X_1,\ldots,X_N$
      - Each variable takes value in finite alphabets, say {0,1}
    - E is set edges capturing "dependencies"
  - Joint distribution:

$$P(X_1 = x_1,\ldots, X_N = x_N) \propto \exp\left(\sum_{i\in V} \phi_i(x_i) + \sum_{(i,j)\in E} \psi_{ij}(x_i, x_j)\right)$$

  - Potentials: weighted independent set model
    - $\phi_i(\sigma) = w_i \sigma$ for $\sigma \in \{0,1\}$ for all $i \in V$
    - $\psi_{ij}(\sigma, \sigma') = -\infty$ if $(\sigma, \sigma') = (1,1)$ and 0 o.w. for all $(i,j) \in E$

Inference: pair-wise graphical model



- Scheduling by sampling as per this distribution is capacity achieving when node potentials/weights = func(queue-size)
  - This view has led to totally distributed Medium Access [Shah-Shin '08 '11]

Inference: pair-wise graphical model

- Find Maximum A Posteriori (MAP) assignment
  - x^* such that

$$x^* \in \operatorname{argmax} \sum_{i \in V} \phi_i(x_i) + \sum_{(i,j) \in E} \psi_{ij}(x_i, x_j)$$

  - In the context of independent set model
    - Solve for maximum weight independent set

$$\begin{aligned} &\text{maximize} && \sum_{i \in V} w_i x_i \\ &\text{such that} && x_i + x_j \leq 1 \text{ for all } (i,j) \in E \\ &\text{over} && x_i \in \{0,1\} \text{ for all } i \in V \end{aligned}$$

    - Computationally hard (to approximate)

Optimization: "prominent" methods

- All optimization methods are highly relevant
  - Linear programming
  - Semi-definite programming
  - Higher-order relaxations
    - Lovasz-Schrijver
    - Sherali-Adams
    - Lassare and Sums-of-squares
  - (Approximate) Dynamic programming
  - …

Graphical model: "prominent" methods

- Graphical model induced techniques
  - Variational inference
    - Mean-field
    - Belief propagation
    - Tree-reweighted and its variations
  - Markov Chain Monte Carlo
    - Gibbs sampling
    - Metropolis-Hastings
    - Simulated annealing
  - …

This talk

- A method for solving graph optimization problems
  - Utilizes graph partitioning
    - Cf. Lipton-Tarjan '80, Baker '94; popularized recently in dist comp
  - Requires nearly linear time computation
  - Makes existing centralized heuristic distributed
- For a large class of graphs (poly-growth, planar, …)
  - Provides good, provably approximation
  - Speeds up computation of existing heuristics
    - Without losing performance
- For any graph, nearly linear time heuristic
  - With easy to evaluate error bound

Algorithm

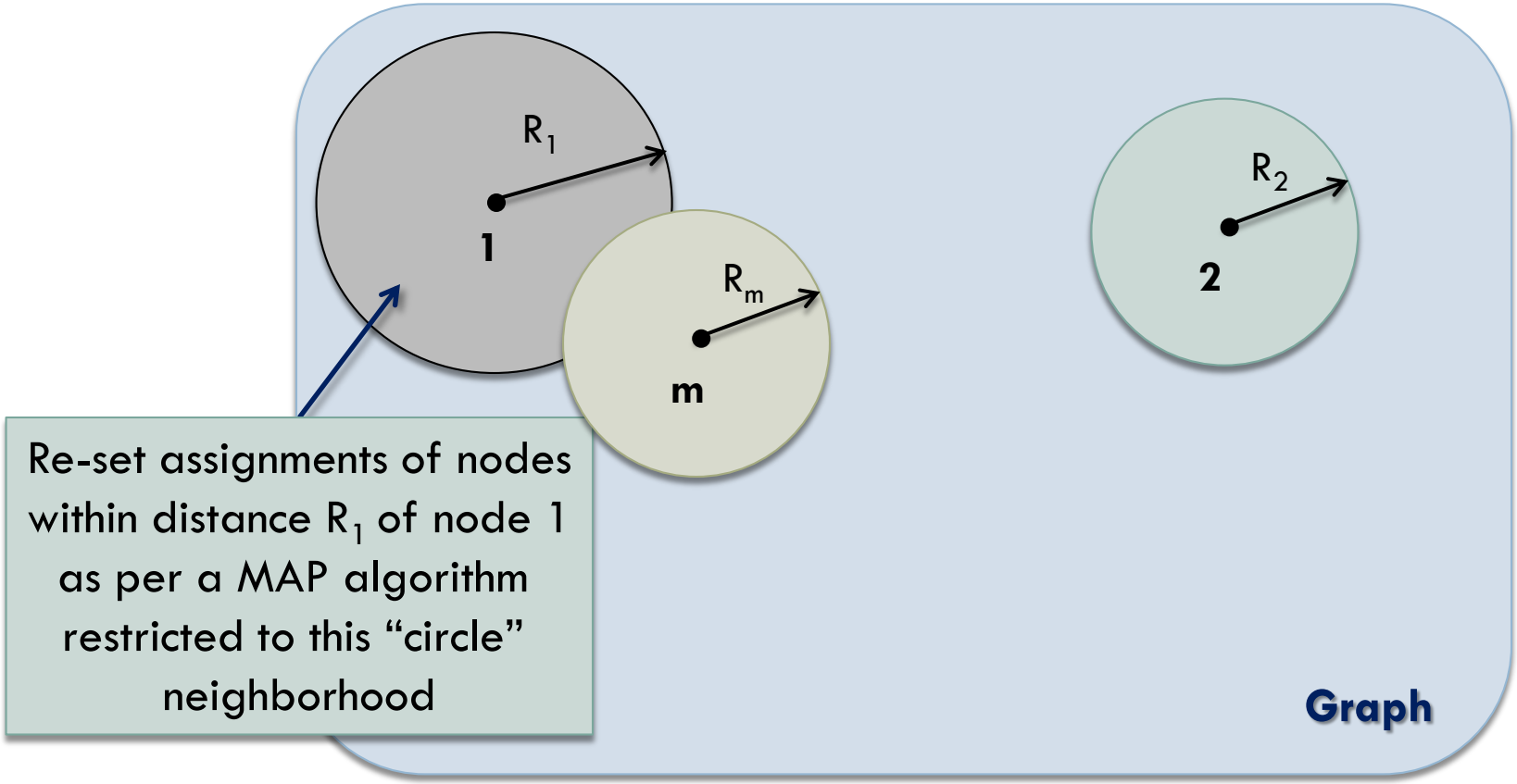
- Two parameters: K and δ

- Useful random variable R
  - Truncated Geometric(ε) with value at most K

$$\mathrm{P}(\mathrm{R}=\ell)=\begin{cases}\delta(1-\delta)^{\ell-1} & \ell<\mathrm{K}\\ (1-\delta)^{\mathrm{K}-1} & \ell=\mathrm{K}\end{cases}$$

Algorithm

- Initialize $X_i = 0$ for all i (or any valid assignment)
- Iteratively, algorithm updates the assignments



Algorithm

- Effectively,
  - Each iteration improves the assignment
    - Restricted to a small random (local) neighborhood
    - By means of possibly exact MAP (exp(K) computation)

- **Theorem 1 (Jung-Shah '08)** Upon selecting all N vertices once (in any order), the algorithms finds (1- ε) MAP approximation for graph with poly-growth rate ρ as long as K scales as $\Theta\left(\frac{\rho}{\varepsilon}\ln\frac{\rho}{\varepsilon}\right)$ (on average, whp upon log N repeats).

Graph with poly-growth

- We call a graph with poly-growth rate ρ if
  - For any vertex v of G and any r
    - N(r,v), number of nodes within graph hops r, is bounded as

$$N(r, v) \leq C\, r^{\rho}$$

for some constant C > 0.

- Example
  - Grid graph, or
  - *Any local* graph betn nodes in a finite dim'l metric space

Algorithm: general graph

- For any graph
  - Exact MAP inside local neighborhood
  - Let E' be set of edges "cut" by the N random partitions
  - Let $C(E') = \sum_{(i,j) \in E'} |\psi_{ij}|_\infty$

- **Lemma 1 (Jung-Shah '08)** Upon selecting all N vertices once (in any order), the algorithms finds MAP approximation whose cost is at most C(E') less than exact MAP for any graph.

Randomized decomposition

- Under our randomized decomposition
  - For graph with poly-growth
  - (and with appropriate choice of K, δ)

Pr(an edge in "cut") ~ε

  - This leads to the desired approximation bound

Algorithm for modularity opt

- Effectively,
  - Each iteration improves the assignment
    - Restricted to a small random (local) neighborhood
    - By means of possibly exact Modularity (super-exp(K) computation)

- **Theorem 2 (Blondel-Jung-Shah '11)** Upon selecting all N vertices once (in any order), the algorithms finds clustering that is within (1- ε) of maximal Modularity for graph with poly-growth rate ρ as long as K scales as $\Theta\left(\frac{\rho}{\varepsilon}\ln\frac{\rho}{\varepsilon}\right)$ (on average, whp upon log N repeats)..

Is it an excellent result?

- Well, looks like it…
  - We've got a linear time PTAS
- But, can we *really* use it
  - Constants scale super-exponentially in $1/\varepsilon$
  - On a 100 node graph
    - DP for ind. set requires $2^{100} \cong 10^{30}$ operations!
    - Modularity opt requires 100^{100} operations!
- Okay, the usual *fantastic* algorithmic result

Algorithm: a *useful* variation

- Use of approximate MAP
  - Suppose inside each "local" neighborhood, we use
    - An approximation MAP algorithm (e.g. Belief propagation)
    - Let it be A(N) MAP approximation (for any graph)
      - with $A(N) = o(N)$ (e.g. $A(N) = \log N$)

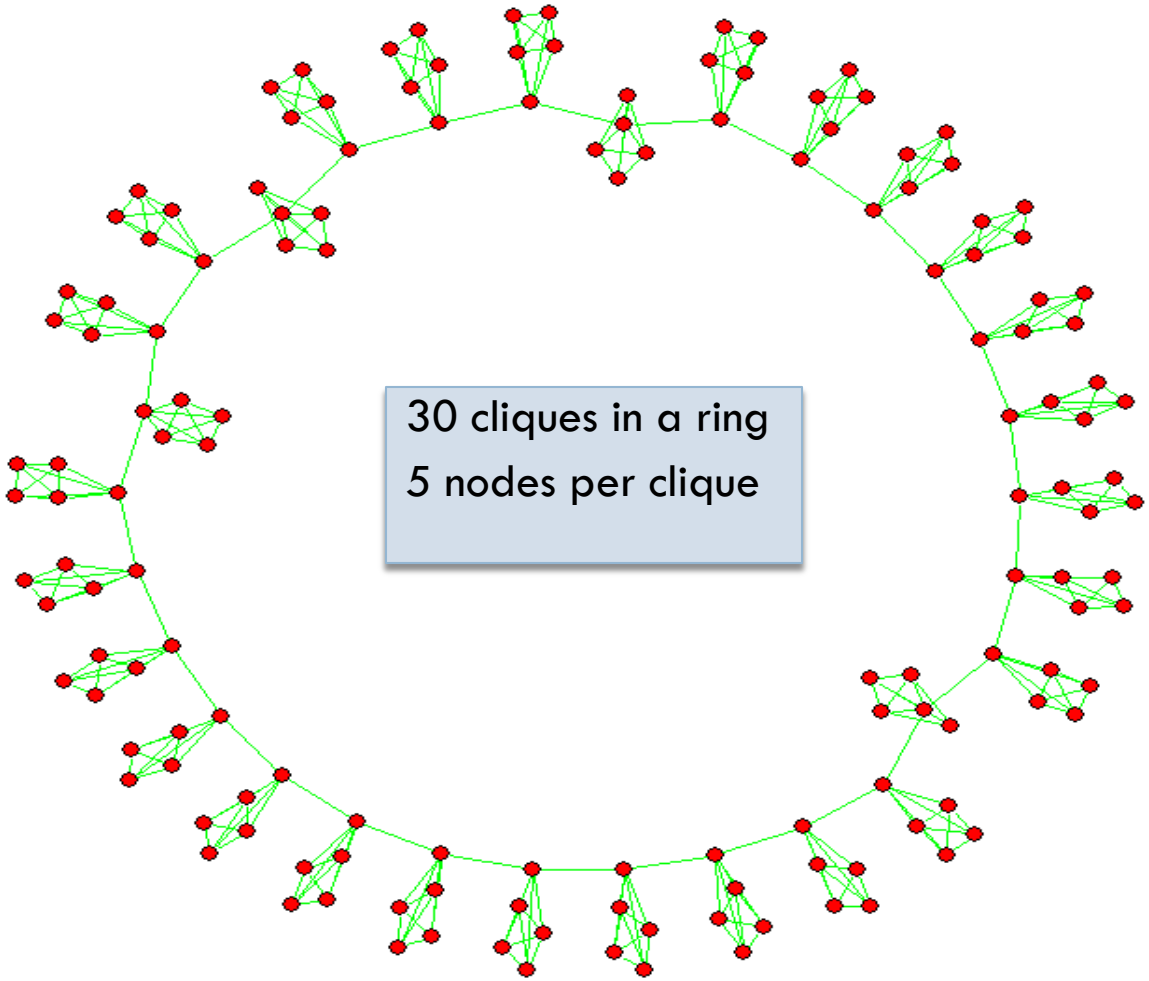
- **Theorem 3 (Jung-Shah '11)** Upon selecting all N vertices once (in any order), the algorithms finds *constant factor* MAP approximation for graph with poly-growth rate ρ as long as K scales as before.

Algorithm: a *useful* variation

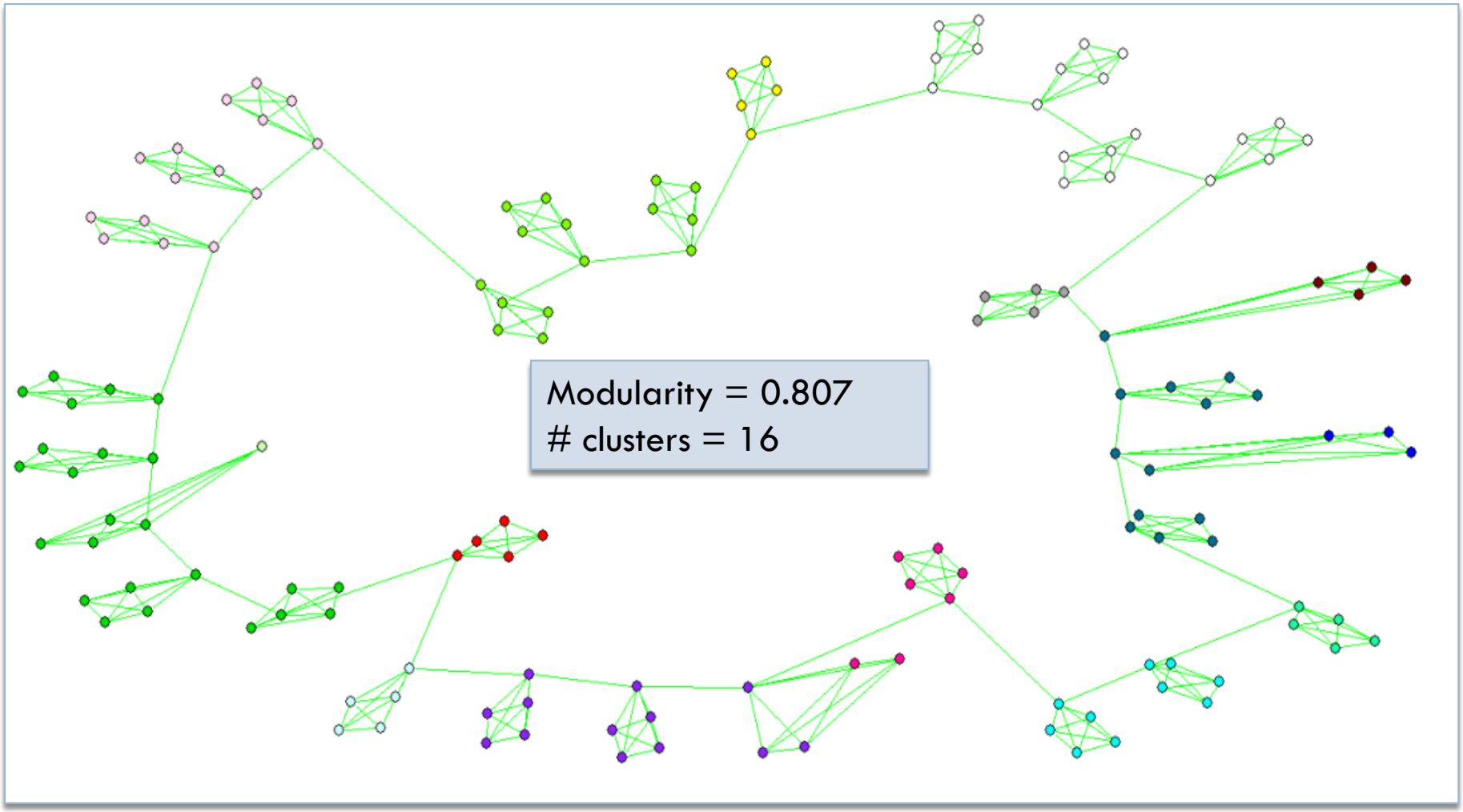
- Use of approximate MAP
  - Suppose inside each "local" neighborhood, we use
    - An approximation Mod Opt algorithm (e.g. Blondel et al '09)
    - Let it be A(N) Mod Opt approximation (for any graph)
      - with $A(N) = o(N)$ (e.g. $A(N) = \log N$)

- **Theorem 4 (Blondel-Jung-Shah '11)** Upon selecting all N vertices once (in any order), the algorithms finds *constant factor* Mod Opt approximation for graph with poly-growth rate ρ as long as K scales as before.

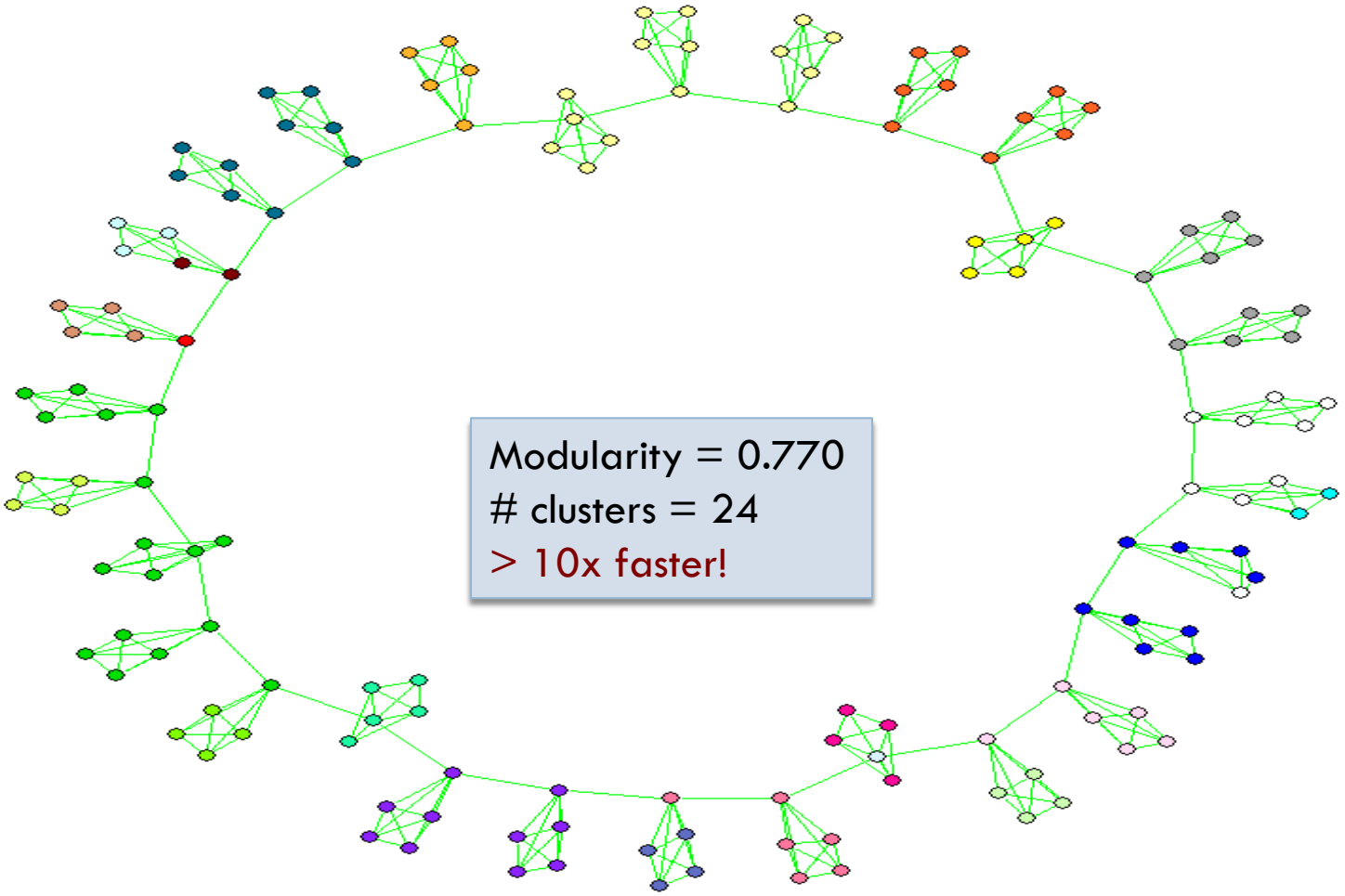
Example



Example: Blondel et al



Example: Blondel et al + partition



Algorithm: variations

- Local property testing (Sudan-Widgerson '98)
  - Does a given graph have certain property P
  - By means of an algorithm that has
    - only error dependent randomized computation cost

- Every monotone graph property is locally testable
  - Alon and Shapira '05
  - Constant: "towering exponential" dependence on $1/\varepsilon$

Algorithm: variations

- Partitioning can be used for property testing
  - Hassidim, Kelner, Nguyen, and Onak '10
  - Reduces the towering dependence to one exponent
  - Homogenous properties

- This algorithm can be used as
  - Local property tester for graphs with poly-growth
  - For various "optimization"-based properties
    - That are not necessarily homogenous

Algorithm: variations

- Algorithm can be used for
  - Estimating log partition function
    - Or normalization constant
  - With guarantees similar to those for MAP
  - It works as a "proof" to establish existence of limit
    - For normalized free-energy
      - for poly-growth graphs with limiting "regular" structure
    - And hence, limit can be evaluated

Summary

- "Meta" algorithm for inference in graphical model
  - For excellent approximation for graphs with geometry
  - Speeds up existing heuristics without losing performance
  - Has easy to evaluate "error" bound for any graph
  - It is effectively the architecture of choice

- Other useful consequences
  - Log partition function (learning)
  - Property testing