Algorithmically Efficient Networks

Kyomin Jung   Devavrat Shah

Laboratory of Information and Decision Systems
Massachusetts Institute of Technology

Motivation

- Algorithms are operational building blocks of a network
    - Scheduling
    - Routing
    - Bandwidth allocation

- Success of a network depends upon
    - Ease of implementing high-performance algorithms

- Many examples of failed successful network design
    - Perfectly designed but *not implementable*

Motivation

- Usually, problems arising in network are algorithmically very hard

- However,
  - Are there simple network structures that allow for
    - Simple algorithm for many of the problems in networks
  - If so, we should try to build networks with such structures

- In this talk, we will search for such network structures

Outline

- Three problems
  - ◦ Packet scheduling (discrete optimization)
  - ◦ Loss probability in network (partition function)
  - ◦ Feasible rate allocation (membership in a convex set)
- Good network structures
  - ◦ Graphs with low doubling dimension
  - ◦ Minor-excluded graphs
    - – For example, planar graph
- Approximation algorithms for three problems
- Discussion

→ Next, an example that will be used throughout the talk

Wireless Network

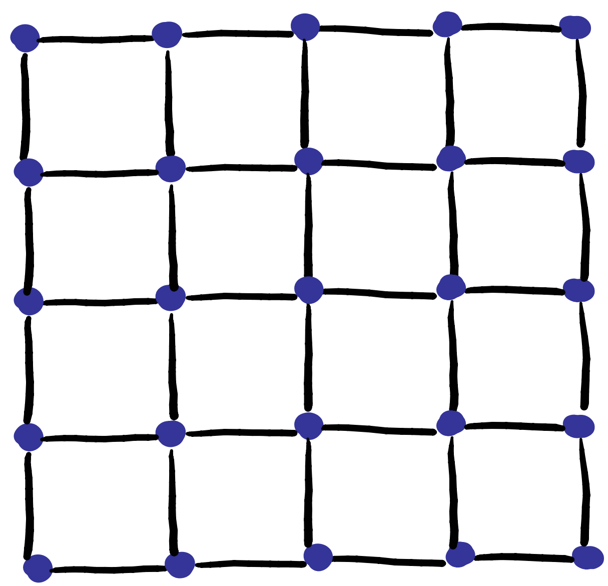
- Consider wireless network of n nodes
  - Graph $G = (V, E)$ with $|V| = n$, and
    - $E = \{(i, j) :$ nodes i and j can communicate$\}$
  - Neighbors of i, $\mathcal{N}(i) = \{j \in V : (i, j) \in E\}$

- Interference model
  - If node i is transmitting then all of its neighbors, that is nodes in $\mathcal{N}(i)$, should not transmit

$\rightarrow$ Set of simultaneously transmitting nodes form an independent set of G

Wireless Network

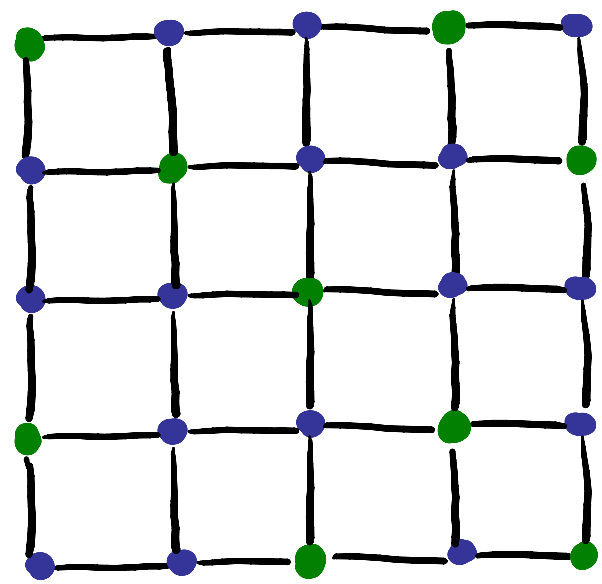
- We will consider grid graph network
    - Only for ease of explanation



5x5 GRID GRAPH

Wireless Network

- Interference in grid graph network

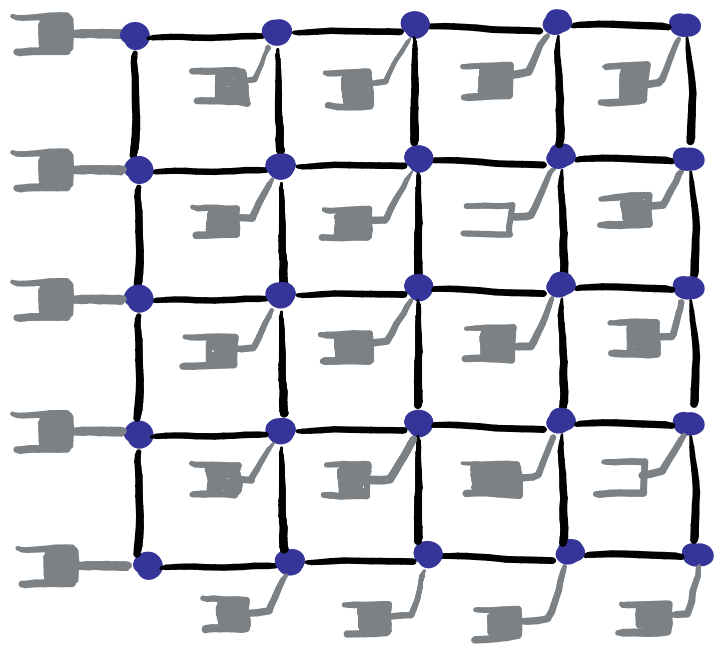


Problem I: Packet Scheduling

- Consider single-hop network
  - Time is slotted denoted by $t \in \mathbb{Z}$
  - Packets of unit-size arrive at nodes of G
    - At rate λ_v for $v \in V$ according an external arrival process
  - Packets are buffered at nodes if required
    - Let $Q_v(t)$ denote queue-size at $v \in V$ at time t
  - Packets depart from network when transmitted
    - That is, it is a single-hop network

- Scheduling algorithm: at each time
  - Choose an independent set of G
    - To schedule transmission of packets at nodes

Problem I: Packet Scheduling



5x5 GRID GRAPH

Notation

- Let $\mathcal{I}(G) \subset \{0,1\}^n$ be set of independent sets of G
  - Let $\Lambda = \text{Co}(\mathcal{I}(G))$

- Capacity region is Λ
  - If $\lambda \notin \Lambda$:
    - Not possible to serve all queues at rate higher than their arrival rate
  - If $\lambda \in \Lambda^o$:
    - Possible to serve all queues at rate higher than arrival rate through a TDMA scheme

$\rightarrow \lambda$ is admissible if $\lambda \in \Lambda^o$

Performance metric

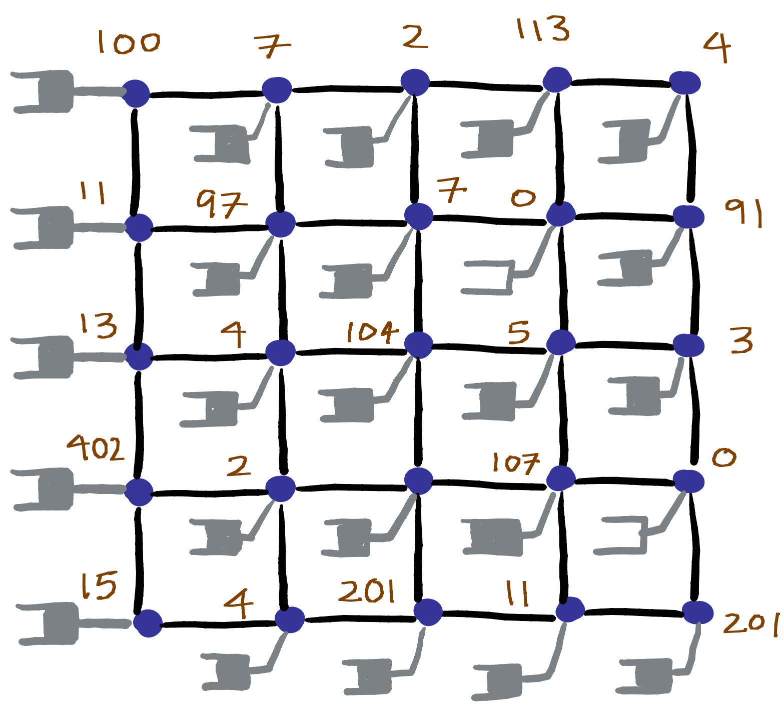
- Throughput
  - ○ Scheduling algorithm is *stable* (delivers 100% throughput), if
    - – For any admissible λ, the average queue-size is finite

$$\sup_t \mathbb{E}\left[Q_v(t)\right] < \infty, \quad \text{for all} \;\; v.$$

- Net average queue-size:
  - ○ $\sup_t \mathbb{E}\left[Q(t)\right]$,
    - – where $Q(t) = \sum_v Q_v(t)$

Maximum Weight Scheduling

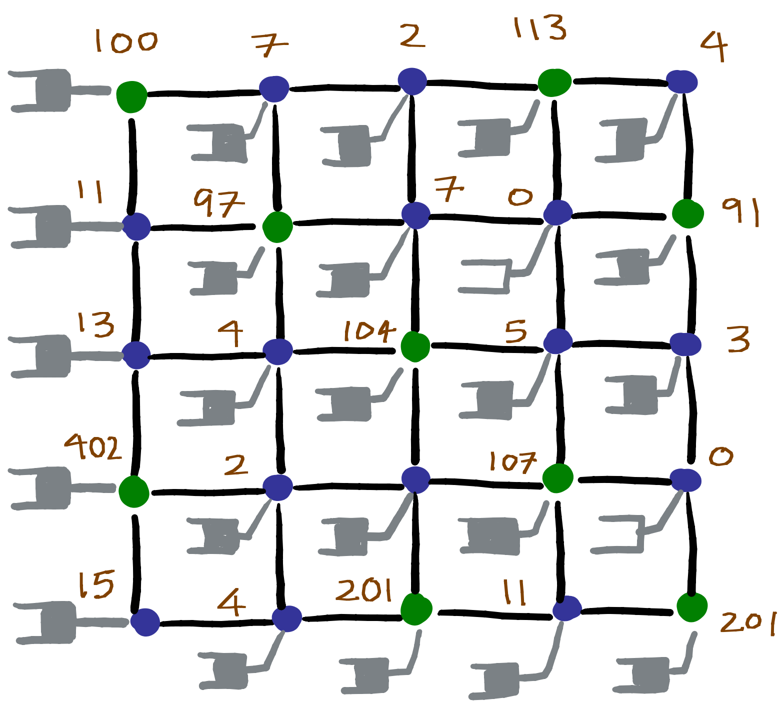
- Consider the node weighted graph G
  - Each node is assigned weight equal to queue-size



5x5 GRID GRAPH

Maximum Weight Scheduling

- Algorithm: max. wt. independent set (MWIS)
  - Every time, choose schedule (independent set) with max. weight,
  - Transfer packets according to this schedule



5x5 GRID GRAPH

Maximum Weight Scheduling

- Tassiulas and Ephremides (1992) proposed this algorithm
  - They showed it to be stable

- It follows that for any G, under max. wt. scheduling
  - The net average queue-size is bounded above as $O(n^2/\varepsilon)$
    - Under Bernoulli i.i.d. arrival process, and
    - $\lambda \in (1-\varepsilon)\Lambda$

Complexity of Max. Wt. Scheduling

- The problem of finding max. wt. independent set is hard
  - ◦ There are instances of weighted graphs such that no polynomial in n time algorithm to find even good approximation unless $P = NP$
- Question: is there any algorithm that is stable
  - ◦ Requires poly in n computation for any graph G
- Answer: Yes
  - ◦ Direct modification of algorithm by Tassiulas (1998)
- Can it be totally distributed ?
  - ◦ Yes, using a *Gossip* mechanism

$\longrightarrow$ So what is the issue ?

Complexity of Max. Wt. Scheduling

- These low (poly in n) complexity distributed algorithms
  - Stable, but
  - Induce large (exponential in n) average queue-size
- Question: is there algorithm that always provides
  - *Small* (poly in n) avg. queue-size for
    - Bernoulli i.i.d. arrival process with
    - $\lambda \in (1 - \varepsilon)\Lambda$ for some fixed $\varepsilon > 0$
  - And has low (poly in n) complexity ?
- Answer: No
  - Under standard computational hypothesis
  - Shah and Tsitsiklis (2006)

$\rightarrow$ We need to search for good graph structures

Problem II: Loss Probability

- Consider network *buffer-less* network
  - That is, *Loss* network
- Packets arrive to node $v \in V$
  - According to a Poisson process of rate λ_v
  - With i.i.d. service requirement of unit mean
- Policy:
  - If packet arrives to a node that can instantly start serving it,
    - Accept it and start transmitting
  - Else,
    - Drop it

Performance

- It is well-known (Everitt-Macfadyen (1983), Kelly (1985))
  - The stationary distribution over $\mathbf{x} = (x_v) \in \mathcal{I}(G) \subset \{0,1\}^n$

$$\Pr(\mathbf{x}) = \frac{1}{Z} \prod_{v \in V} \lambda_v^{x_v},$$

where normalization constant Z is

$$Z = \sum_{\mathbf{x} \in \mathcal{I}(G)} \prod_{v \in V} \lambda_v^{x_v}.$$

- We are interested in loss-probability, that is
  - What fraction of packets get dropped ?

Performance

- PASTA property
  - ◦ Arrivals see time-stationary average
  - $\rightarrow$ Compute time-stationary acceptance probability

- Packet arriving at node u is accepted iff
  - ◦ Node u is empty and
  - ◦ All neighbors of u are empty
  - $\rightarrow$ $\Pr(\text{acceptance}) = Z_u/Z$, where

$$Z_u = \sum_{\substack{\mathbf{x}\in\mathcal{I}(G):\\ \mathbf{x}_{\mathcal{N}(u)}=0}} \prod_{v\in V} \lambda_v^{x_v}.$$

Performance

- Thus, to evaluate acceptance probability (equivalently loss prob.)
  - ◦ Sufficient to compute *partition* function Z
    - – That is, sum of all weighted independent sets

- This problem is computationally hard in general setup
  - ◦ For example, Dyer, Frieze and Jerrum (1999) showed hardness for counting independent sets in graphs with degree more than 25

$\longrightarrow$ We need simple network graph structures

Problem III: Feasible Rate Allocation

- Consider rate allocation to links of G
  - $\mu = [\mu_e]_{e \in E}$

- Is there a TDMA scheme so that
  - Rates μ can be allocated to edges while satisfying the communication constraints
  - If yes, obtain a TDMA decomposition of μ

- This is an essential *sub-routine* for many questions
  - For example, in the optimization based cross-layer design

Problem III: Feasible Rate Allocation

- This question is equivalent to
  - Membership query for a convex set

- This question is known to be computationally hard
  - E. Arikan (1984)

- Again,
  - Can we identify simple graph structures ?

Rest of The Talk

- Good graph structures
  - Graphs with low doubling dimension
  - Minor-excluded graphs

- Algorithms for graph with good structure
  - Max. Wt. Ind. Set
    - relevant previous results
  - Log-partition function
  - Membership query

- Discussion

Good Graph Structure

- Given a (family of) graph $G = (V, E)$
  - Vertices V and edges E

- Let $\mathcal{B} \subset V$ be a random set such that
  - $\Pr(v \in \mathcal{B}) \leq \varepsilon$ for any $v \in V$
  - $G' = (V \backslash \mathcal{B}, E')$ is made of connected components of diameter Δ

  $\rightarrow$ Called (ε, Δ)-decomposition of G

- A graph is *good* if it admits (ε, Δ)-decomposition
  - For any $\varepsilon > 0$ and
  - Δ, possibly dependent on ε, but independent of $n = |V|$

- Next, two important examples

Good Graph Structure: Example I

- A graph G has doubling dimension ρ if
  - Graph neighborhood of any $v \in V$ of radius r, say $\mathbf{B}(v, r)$ is s.t.

$$\mathbf{B}(v, r) \subset \cup_{k=1}^{K} \mathbf{B}(v_k, r/2),$$

  for some $K \leq 2^\rho$ neighborhoods of radius $r/2$

- Equivalently, for any $v \in V$
  - $\mathbf{B}(v, r) \leq (2r)^\rho$ for $r \geq 1$
  - That is, *polynomial growth* of neighborhood

Good Graph Structure: Example I

- Given G with doubling dimension ρ
  - A simple & explicit algorithm for finding (ε, Δ)-decomposition
    - where, $\Delta = \frac{24\rho}{\varepsilon} \log\left(\frac{48\rho}{\varepsilon}\right)$.

- Notation: let $\mathbf{Q}$ be random variable on $\{1, \ldots, \Delta\}$

$$\Pr(\mathbf{Q} = i) = \begin{cases} \varepsilon(1-\varepsilon)^{i-1} & \text{if } 1 \le i < \Delta \\ (1-\varepsilon)^{\Delta-1} & \text{if } i = \Delta. \end{cases}$$

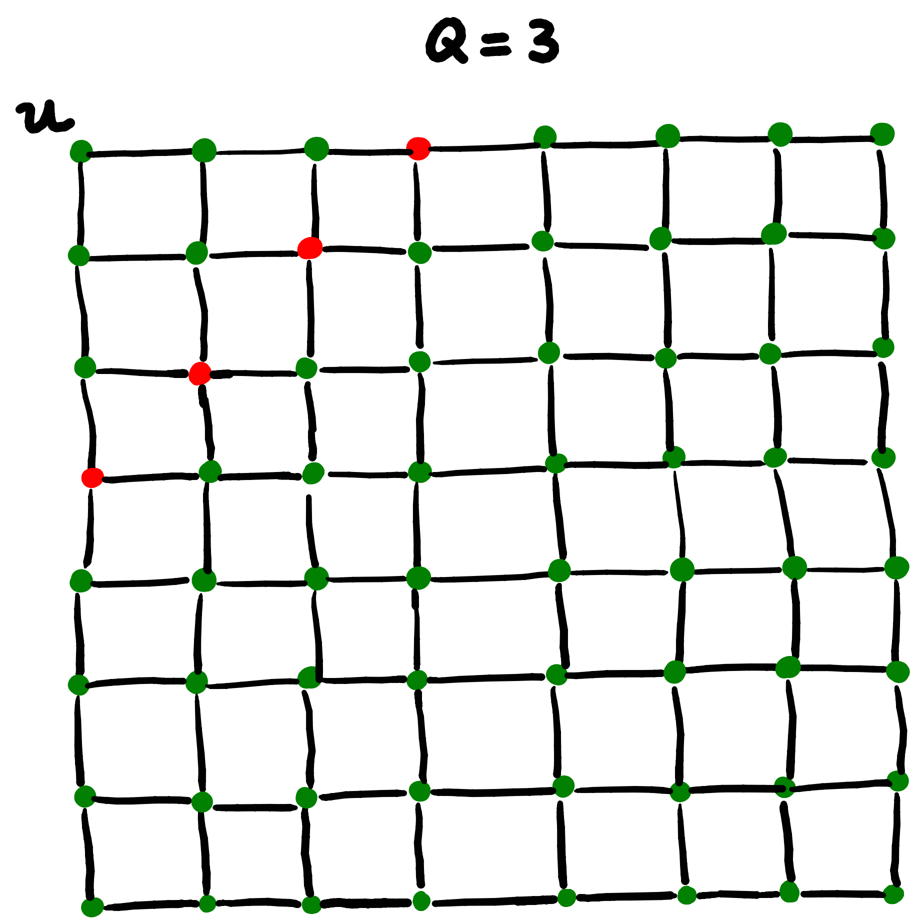
- Next, we describe the algorithm that uses distribution of $\mathbf{Q}$

Good Graph Structure: Example I

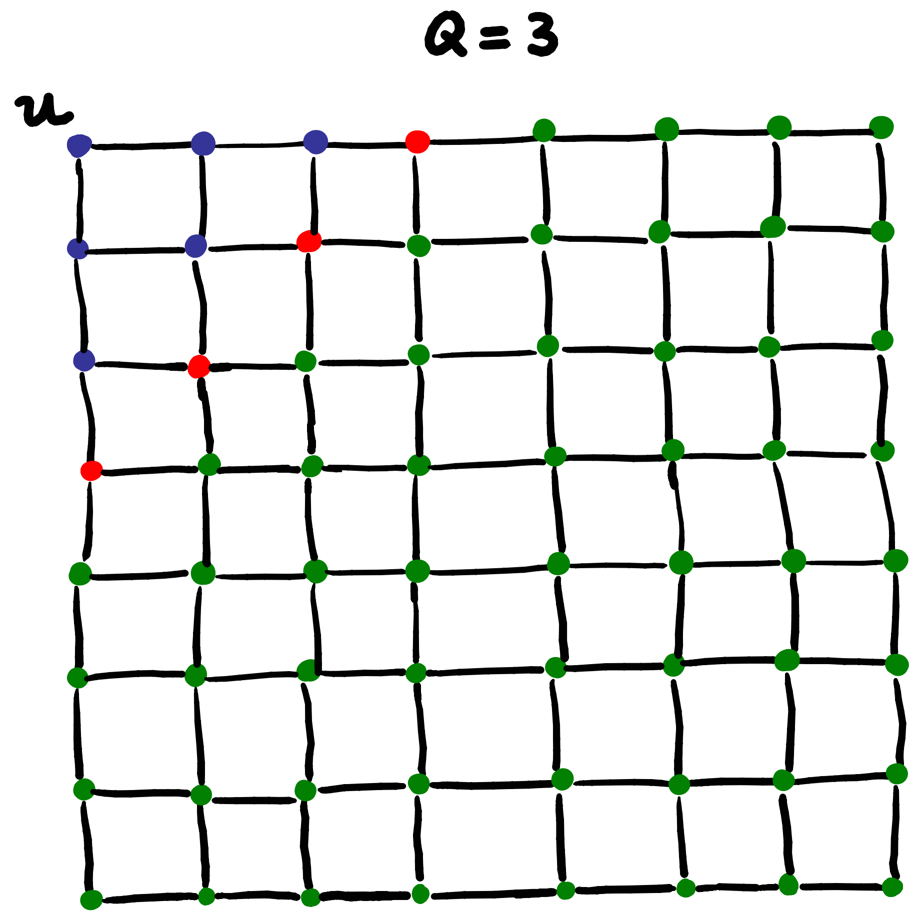
- Decomposition algorithm
  - Initially, all nodes are colored green
  - Iteratively, color them red and blue as follows
  - Choose a green node, say u, arbitrarily and sample number from $\{1, \ldots, \Delta\}$ as per distribution of $\mathbf{Q}$ independently
    - color all green nodes at distance $\mathbf{Q}$ from u as red
    - color all green nodes at distance less than $\mathbf{Q}$ from u as blue
  - Repeat the above till no more green node is left
  - Output red nodes as $\mathcal{B}$

- An example of algorithm's iteration

Good Graph Structure: Example I



Good Graph Structure: Example I



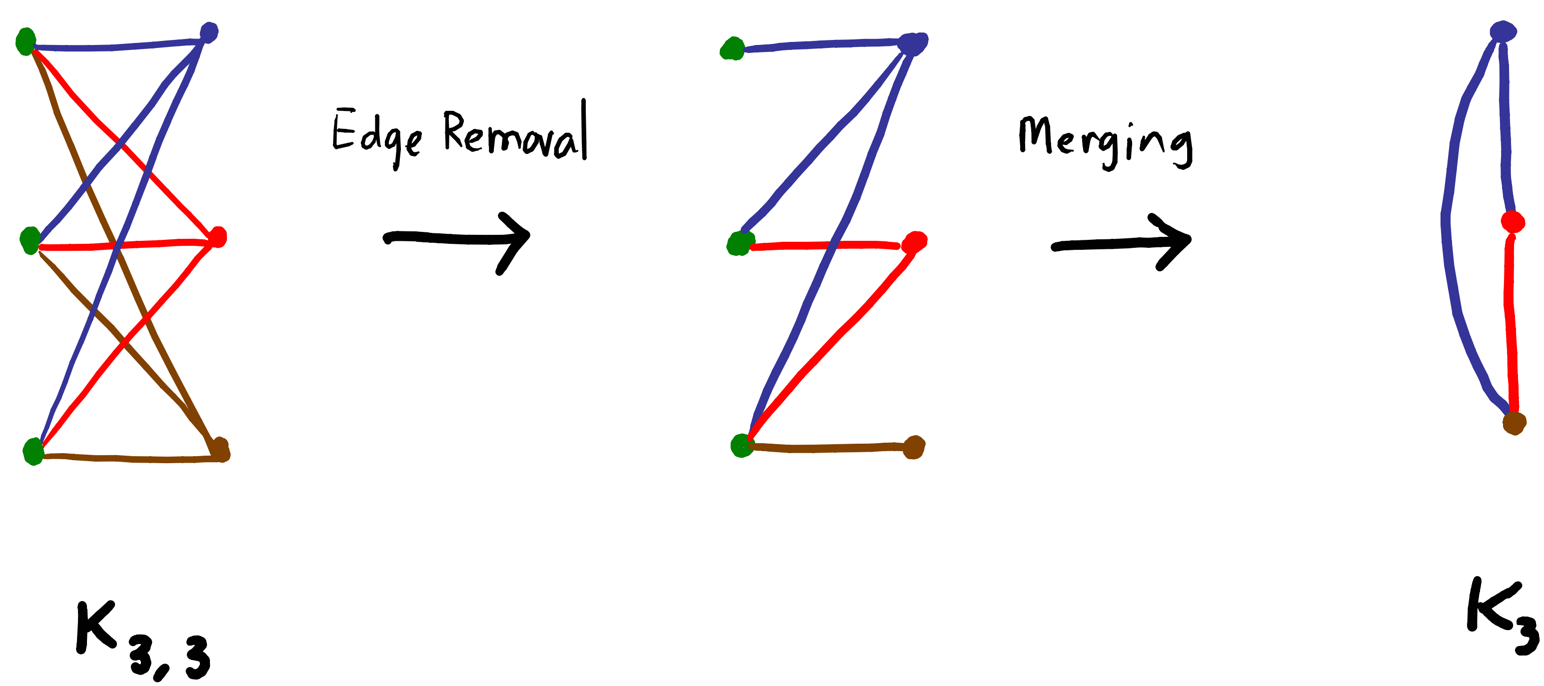
- **Lemma.** The above algorithm produces (ε, Δ)-decomposition of any graph G with doubling dimension ρ.

Good Graph Structure: Example II

- Graph H is a *minor* of G if
  - H can be obtained from G through an arbitrary sequence of the following two operations:
    - removal of edge
    - merging of two connected vertices

Good Graph Structure: Example II

- Example: K_3 is a minor of $K_{3,3}$



Good Graph Structure: Example II

- Graph H is a *minor* of G if
  - ○ H can be obtained from G through an arbitrary sequence of the following two operations:
    - removal of edge
    - merging of two connected vertices

- A family of graph that exclude certain finite sized graph as their minor are *good graph structure*
  - ○ For example, planar graph
    - excludes $K_{3,3}$ and K_5 as its minor

- Next, an explicit construction due to Klein, Plotkin and Rao (1994)

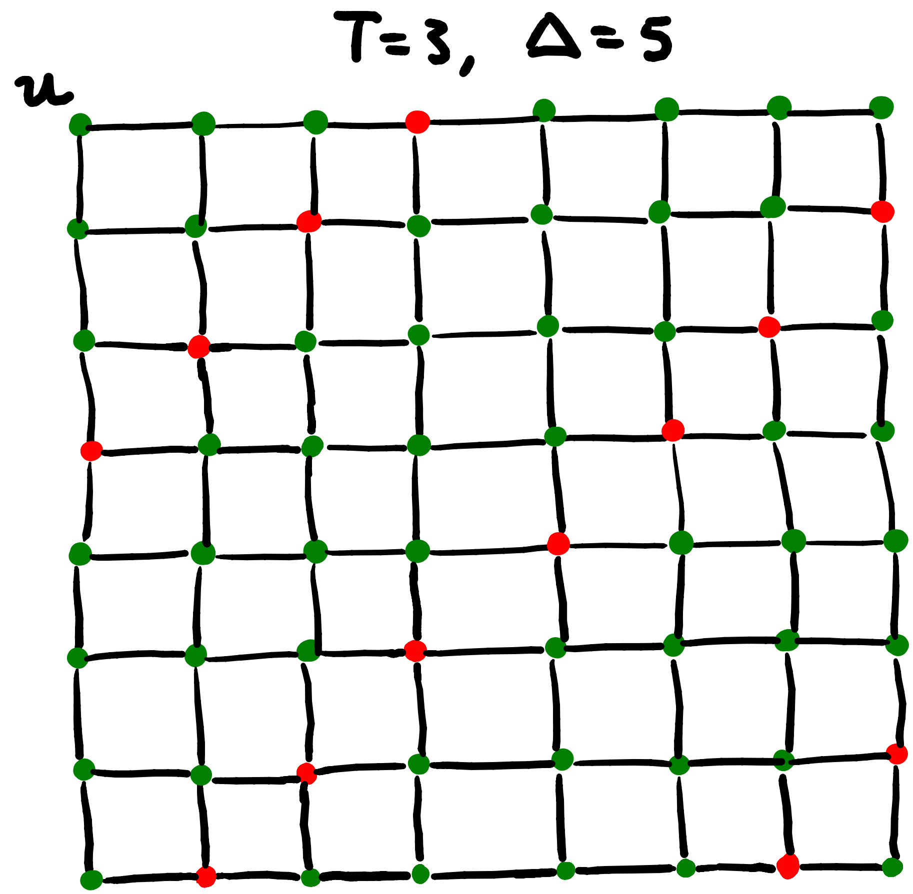
Good Graph Structure: Example II

- Let G exclude graph H of r nodes as a minor
  - ◦ Algorithm for (ε, Δ)-decomposition scheme
    - – For any $\varepsilon > 0$ and $\Delta = 2r/\varepsilon$

- Decomposition algorithm
  - ◦ Initially, all nodes are colored green
  - ◦ For $j = 1, \ldots, r$ do the following:
    - – in each connected component of green nodes, choose an arbitrary node u
    - – if all nodes are at distance at most Δ then stop
    - – else, sample number $\mathbf{T}$ from $\{1, \ldots, \Delta\}$ unif. at random color nodes at distance $\mathbf{T} + k\Delta, k \geq 0$ from u as red
  - ◦ Output red nodes as $\mathcal{B}$

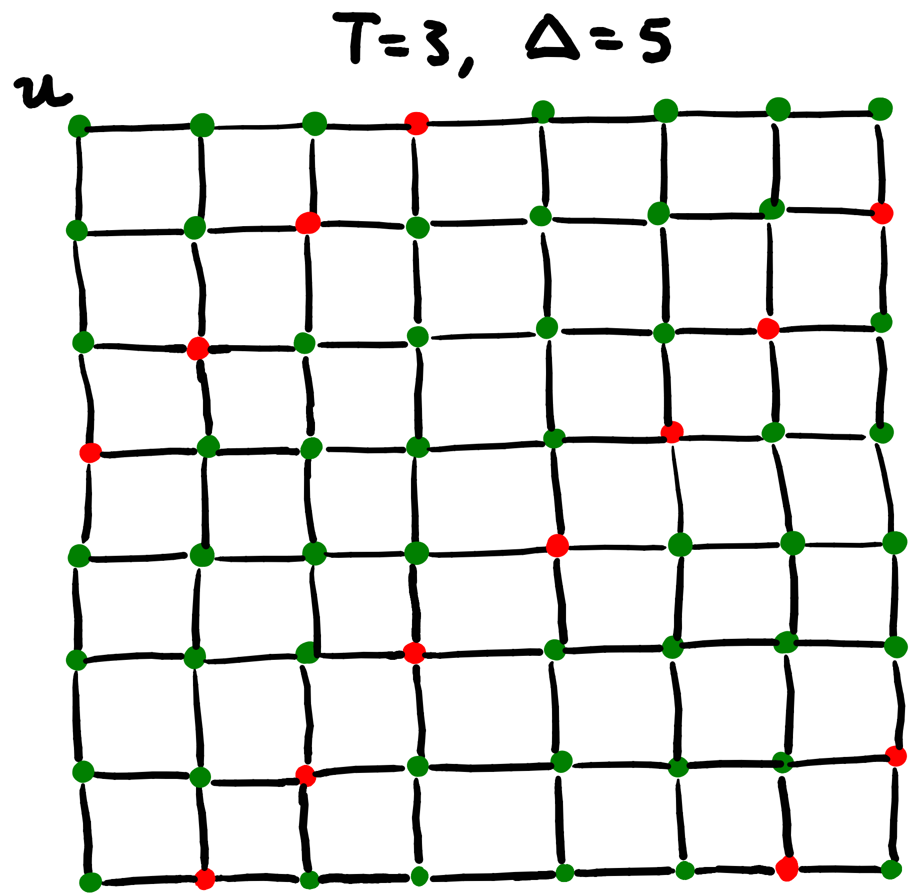
Good Graph Structure: Example II

- Example of an iteration of the algorithm



Good Graph Structure: Example II

- Example of an iteration of the algorithm



- **Lemma. [KPR94]** The above algorithm produces (ε, Δ)-decomposition of any graph G with doubling dimension ρ.

Good Graph Structure and Algorithm

- Good graph sturcture admits (ε, Δ) decomposition
  - Explicit construction for graphs that have
    - low doubling dimension
    - minor exclusion

- Next, algorithms for three problems
  - That use (ε, Δ)-decomposition for finding good solution

Some History

- Optimization algorithm using decomposition
  - Started Lipton-Tarjan (1977) for max. size independent set for planar graph using planar separator theorem
  - Baker (1994) extended this for max. weight independent set for planar graph
  - Hunt et. al. (1998) extended it for *disk graphs*
  - Nieberg et. al. (2004) gave deterministic algorithm for disk graphs
  - Kuhn (and co-authors) (2005) extended for graphs with low-doubling dimension but restricted to max. size independent set
- Recent results using such algorithms for near optimal throughput scheduling, e.g.
  - Balakrishnan et. al. (2003), Sharma et. al. (2006), Sarkar et. al. (2007)

Some History

- There are no algorithms for max. wt. independent set
  - Graphs with low doubling dimension, and
  - Minor excluded graphs

- Next, the algorithm

Max. Weight Independent Set

- Given G with non-negative node weights,
  - Obtain (ε, Δ)-decomposition $\mathcal{B}$
  - Set all nodes in $\mathcal{B}$ to 0
  - Let $S_1, \ldots, S_\ell$ be connected components of $G' = (V \backslash \mathcal{B}, E')$
    - compute max. wt. independent set restricted to each S_i
  - Output thus computed assignment of nodes as estimate of max. wt. indepednet set

- To obtain, high probability of good approximation, repeat the above $\Theta(\log n)$ times and output the max. weighted estimate

Max. Weight Independent Set

- **Theorem.** The algorithm produces independent set whose weight is at least $(1 - 2\varepsilon)$ times the weight of max. wt. independent set with probability at least $1 - 1/\text{poly}(n)$.

- For any $\varepsilon > 0$ (not scaling with n), for running time to be poly in n, we require that
  - The $\rho = o(\log\log n / \log\log\log n)$, for doubling dimension graphs
  - The degree be bounded as $o(\log\log n)$ for minor excluded graphs

- Derandomization
  - The decomposition schemes can be made deterministic as they access less than $O(\log n)$ bits of randomness

Approximate Log-partition

- Some history
  - ○ Popular method is based on Markov chain monte carlo approach
    - – Starting work by Jerrum and Sinclair (1989)
  - ○ Recent result by Weitz (2006) about counting independent set
    - – for any graph with degree at most 5 in a deterministic manner
  - ○ For planar graph, Hayes (2006) showed independent sets can be counted upto a bit higher degree constraint

$\rightarrow$ All require proving strong mixing property (in time or space)

- We will obtain approximate $\log Z$
  - ○ Requires only *good graph structure* property

Approximate Log-partition

- Algorithm
  - Obtain (ε, Δ)-decomposition $\mathcal{B}$
  - Remove all nodes in $\mathcal{B}$ from G to produce G'
  - Let $S_1, \dots, S_\ell$ be connected components of $G' = (V \backslash \mathcal{B}, E')$
  - For each S_i, compute
    - Partition function $Z(i)$ that is restricted to S_i
  - Let, $\log \bar{Z} = \sum_{i=1}^{\ell} \log Z(i)$
  - Repeat $K = \Theta(\log n)$ times: $\log \bar{Z}_j, 1 \le j \le K$
  - Declare estimate $\log \hat{Z}$, where

$$\log \hat{Z} = \frac{1}{K} \sum_j \log \bar{Z}_j.$$

Approximate Loss Probability

- **Theorem.** The algorithm produces output $\log \hat{Z}$ such

$$(1-\varepsilon)\log Z \le \log \hat{Z} \le (1+\varepsilon)\log Z,$$

for any $\varepsilon > 0$ with probability at least $1 - 1/\text{poly}(n)$.

- For any $\varepsilon > 0$ (not scaling with n), for running time to be poly in n, we require that
  - The $\rho = o(\sqrt{\log\log n})$, for doubling dimension graphs
  - The degree be bounded as $o(\sqrt{\log\log n})$ for minor excluded graphs

- Again, algorithm can be made deterministic

Approximate Rate Allocation

- Given μ, we will obtain
  - Approximation of its decomposition if $(1+\varepsilon)\mu$ is feasible
  - Else, declare infeasible if $(1-\varepsilon)\mu$ is not feasible
- Algorithm
  - Obtain (ε, Δ)-decomposition $\mathcal{B}$
  - Remove all nodes in $\mathcal{B}$ from G to produce G'
  - Let $S_1, \ldots, S_\ell$ be connected components of $G' = (V \backslash \mathcal{B}, E')$
  - Check feasibility of μ restricted to S_i
    - If not feasible, generate answer *infeasible*
    - Else, $\bar{\mu}(i) = \sum_m \alpha_m^i I_m^i$ be decomposition of μ w.r.t. S_i
  - Let, $\bar{\mu} = \sum_{i=1}^{\ell} \bar{\mu}(i)$
  - Repeat $K = \Theta(\log n)$ times : $\bar{\mu}_j, 1 \leq j \leq K$
  - Declare *feasible* with estimate $\hat{\mu} = \frac{1}{K} \sum_j \bar{\mu}_j$.

Approximate Rate Allocation

- **Theorem.** For any $\varepsilon > 0$, with probability at least $1 - 1/\text{poly}(n)$, the algorithm produces the following output:
  - *infeaseible*, if $(1-\varepsilon)\mu$ is infeasible.
  - *feasible*, if $(1+\varepsilon)\mu$ is feasible, the decomposition $\hat{\mu}$ such that $\hat{\mu} \leq \mu \leq (1-2\varepsilon)^{-1}\hat{\mu}$

- For any $\varepsilon > 0$ (not scaling with n), for running time to be poly in n, we require that
  - The $\rho = o(\log\log n / \log\log\log n)$, for doubling dimension graphs
  - The degree be bounded as $o(\log\log n)$ for minor excluded graphs

- Again, algorithm can be made deterministic

Discussion

- We have seen good graph structure for networks
  - Allow for tractable algorithm design for three important questions

- What next ?
  - Making algorithms simpler and closer to implementation
  - Understanding structure of models of networks, e.g.
    - Preferential connectivity model for Internet
  - Other problems