PODC 2018

David Ellis Hershkowitz

July 29, 2018

These are the notes I took while at PODC 2018. I mostly attended talks on message-passing models and I've omitted the talks for which my notes were not great. These notes were written while trying to keep up with the talks and so are not free from errors. Cheers!

Contents

1 July 24, 2018

1.1 Themis Gouleakis on Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover

MPC model defined

MPC Vs PRAM

MPC:
Memory access: partitioned
Local computation: free

PRAM:
Memory access: arbitrary
Local computation: expensive

MIS
This work: give a $O(\log\log\Delta)$-round algorithm for MIS in MPC for $S = \theta(n)$ (memory of each machine)

Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover

Sequential algorithm: greedily put nodes in the MIS

How to simulate in MPC? Choose a partition Define chunk $V_k := [\frac{n}{\Delta^{\alpha^{k-1}}}, \frac{n}{\Delta^{\alpha^k}})$ For each chunk
a) Machine 1 recives all $G[V_k]$ edges
b) Broadcast $MIS((G[V_k])$
c) Update local memories

Do this until the max degree at most $\log^{10} n$
At this point run the CONGESTED CLIQUE algorithm that uses $O(\log\log\Delta)$ rounds

The 1st phase can be simulated in CONGESTED CLIQUE

Maximum Matching and VC
This work: $(1+\epsilon)$-approximate maximum matching and $(2+\epsilon)$-approximate vertex cover in $O_\epsilon(\log\log n)$ rounds where $S = \tilde{O}(n)$

Their MPC algorithm simulates the following LOCAL algorithm

$F \subseteq V$ is frozen vertices Initially assign weight $\frac{1}{n}$ to every edge For every vtx if the sum of weights of incident edges is close to 1, freeze that vertex. Then for all other non-frozen edges increase weight by $1/(1-\epsilon)$

Key Technical Ideas
Is there an MPC simulation in $O(\log\log n)$ rounds?
Do "random vertex partitioning"
"Random threshold"

1.2 Michal Dory on Distributed Spanner Approximation

Gave multiplicative spanner definition
Most work in spanners gives universal gurantees for all graphs

What about approximating?
Much work in centralized setting
This talk on the distributed setting

Main result: $(1+\epsilon)$ approximation in the LOCAL model

What about CONGEST?
Give hardness result for directed or weighted graphs
This gives a strict separation between CONGEST and LOCAL

Need about $\Omega(\sqrt{n/\alpha})$ for an α-approximation for randomized algorithm and directed and $\Omega(n/\sqrt{n})$ for deterministic
Show the reductions from communication complexity
Build a graph G with a sparse spanner iff the inputs to the CC problem have a certain property

Using set disjointness so need change of a single input bit changes many eges of the spanner

Graph construction: Have blocks each of which is conneted to one vertex outside the block. Every input bit to CC affects the distance between every two vertices.

2-Spanners
Dinitz et al.: Give a $O(\log n)$ round and $O(\log n)$ approximation in expectation using only polynomial local computations
Can we give an $O(\log(m/n))$-approximation? Can we guarantee the apx ratio?
Defines stars and densities; intuition: choose a dense star

Gave Kortsarz and Peleg greedy algorithm for 2-approximation
How to make distributed? Could just simulate but don't want to do communication in the whole graph. Also would like to add many stars in parallel.

Find locally dense stars, add edges to the spanner. However, now you may add too many stars in parallel; to overcome this first find stars densest in 2-neighborhood; each candidate chooses a random numberl uncovered edges vote for the first candidate that covers it; a star is added if it gets at least $\frac{1}{8}$ of the votes of the edges it covers.

Future Work:
Is it possible to show a CONGEST algorithm?

1.3 Michal Dory on Distributed Approximation of Minimum k-edge connected Spanning Subgraphs

Gave $O(D)$ rounds for 2-approximation for $k=2$. For $k=3$ $O(D\log^3 n)$. A general $O(knD)$ round for $O(\log k)$

This work

1. $k=2$ and $\tilde{\{O\}}(D+\sqrt{n})$ round $\log n$ approximation
2. Another result I missed

General idea: augment connectivity gradually; i.e. augment $k-1$ to k

Solving the augmentation problem
Goal is to cover all cuts of size $k-1$ with a minimum weight set of edges

Define the cost-effectiveness of an edge as the number of cuts divided by its weight
If just greedily choose according to cost-effectiveness get an $O(\log n)$ apx a la set cover.
But this is inherently sequential
We would like to add edges simultaneously; So how to break symmetry and how to compute cost-effectiveness?

For $k = 2$ start with a spanning tree then augment to 2-connectivity; so need to cover all tree edges.
Algorithm is similar to that of previous talk: All candidates are edges with max effectiveness are candidates.
Each tree edge votes for the first candidate that covers it. An edge is added if at least $\frac{1}{8}$ of the tree edges it covers votes for it

High level idea: decompose tree into fragments

Higher values of k
In general edges in a cut could be far away so not clear how to compute cost-effectiveness
Use "cycle space sampling technique" to compute cost-effectiveness
Each non-tree edges chooses a label and each edge is labeled with the xor of the tree edge labels that cover it
Two edges define a cut $\leftrightarrow$ they have the same label

After computing cost-efectiveness show a way to break symmetry

Open Questions
Sublinear algos for $k > 2$ for weighted k-ECSS

1.4 Christian Konrad on Brief Announcement: Distributed Minimum Vertex Coloring and Maximum Independent Set in Chordal Graphs (brief announcement)

Talk is about min vertex coloring in LOCAL Color a graph; want to find a $\chi(G)$-coloring where that's the chromatic number.
NP-hard to approximate within $n^{1-\epsilon}$
But, in distributed setting can use network decompositions to get a $O(\log^2 n)$ round algorithm
So they're interested in can you get a constant factor approximation

Chordal graph: every cycle of at least 4 vertices contains a chord

Their Results
$(1 + \epsilon)$-approximation in $O(\frac{1}{\epsilon} \log n)$rounds
Lower bound: $\Omega(\frac{1}{\epsilon}) + \log n$ lower bound
Main technique: tree decompositions. Show every node in a distributed setting can get a local view of a tree decoposition; then get an interval graph and color optimally

1.5 Magnus M. Halldorsson on Simple and Local Independent Set Approximation

IS is fundamental
Worthwhile to examine the approximability
Classic bound of Turan says every graph contains a large enough IS
Improved by Caro and Wei

Boppanas Proof
Permute vertices at random. Let I be all vertices whose neighbors all appear later in the order

Gave proof of Caro and Wei bound

Easy to turn into a distributed algorithm.
Also a streaming algorithm and online algorithm

Analyze the approximation algorithm (compared to trivial Δ-approximation).
Then looked at weighted graphs

Gave modified algorithm which is comparable to previous result (though theirs is in expectation); this can be viewed as a single round of Luby's algorithm

2 July 25, 2018

2.1 Michael Elkin on Near-Optimal Distributed Routing with Low Memory

Problem has 2 phases: preprocessing and routing

In preprocessing compute a label and table
Once table and label computed a routing phase takes place where u must route message M to v

Based on table and label vertex must compute the next hop to send to

$P = (u = x_0, x_1, x_2, \ldots, v = x_h)$ is the routing path

The routing is *correct* if the path ends in v
The stretch of the routing path is the max over pairs of vertices of how much their route is longer than their distance in the graph; an important parameter

Also table and label sizes are important

Basic Tradeoff for routing centralized constructions studied by previous work in the centralized setting

So how can we efficiently compute tables and labels: number of rounds in CONGEST needed and memory requirements of vertices

Some previous work on this by Awerbuch, Bar-Noy, Linial and Peleg but their memory requirement was small average memory

Previous work of Elkin and Neiman gave similar table and label sizes but large construction time

Thier result: label and table size is same as in previous work but better construction time, namely $D+n^{.5+1/k}$

Also introduce the **CONGEST-RAM Model** where one is able to send an edge weight in a single message regardless of its weight

Basic Idea in Previous Schemes:
Sample a set V' of size about $\sqrt{n}$ and build $G' = (V', E', \omega')$ (dfn of E' and ω' given)

Defined a (β, ϵ)-hopset $G'' = (V', E'', \omega'')$ for G' where $\forall x', y' \in V'$ we have the distance in G of x' and y' is at most their β distance in $G' \cup G''$ ...

Build hopset for a graph without ever fully constructing the graph itself

Tree Routing
Want to build tables and labels for a tree T in about $D + \sqrt{n}$ rounds. Gives a new distriubted tree routing algorithm.

Open Problems

1. Close gap for distributed routing in general graphs
2. Better distributed routing for restricted graph families
3. Would like to get stretch of $4k - 5$

2.2 Assaf Yifrach on Fair Leader Election for Rational Agents in Asynchronous Rings and Networks

Motivation
Often in distributed algos first a preprocessing leader election step
However, sometimes nodes have a preference over the elected leader
Their goal is to elect a leader uniformly without allowing bias
Consider this problem on a ring

Outline
Problem by example Basic protocol given State their results Give main concept

Problem
3 nodes, 0, 1 and 2. Could just let node 0 elect a random leader and then forward results. However, if node 0 is biased in can easily change the result.
Another approach: let each node randomize the leader and output the sum modunlo 3. But still a single node can cheat; it could wait to receive selections of other nodes before sending its own selection

Clarifications
Adversaries succeed only if they can make honest processors believe the protocol completed properly
A single adversary could just stop sending messages but the protocol will not finish

Solution for n=3 Nodes
Suggested by Abraham et al.

Each node chooses a d_i uniformly and the leader is the sum mod 3 but we change how we share values. Force nodes to commit. Node 0 executes as before but nodes 1 and 2 act as buffers. Conceptually the procol executes in "rounds." At the end every node has received every d_i; also nice property that at end every node receives its own value; so make each node validate that its last incoming message is equal to its first outgoing message.

In this protocol node 1 cannot cheat

Motivation
So what is this work looking for? Want to give resilience to as many adversaries as possible

Assumptions
No side-channel communication adversaries
Unbounded adersary communication
Communication is asynchronous

Their Results Analyzing Past Protocol for Multiple Adversaries
For $\sqrt{n \log n}$ randomly located adversaries can select leader
For $n^{1/3}$ adveraries that are adverarily located can elect leader
Resilience to $n^{1/4}$ adversaries

New Protocol
Reslience to $\sqrt{n}$ adversaries

In an "honest exectution" all nodes are always on the same "round"

2.3 Suman Sourav on Leader Election in Well-Connected Graphs

Two well-known variants of leader election:

1. Explicit Leader: nodes know who leader is

2. Implicit Leader: nodes know if they are leader

Other variants: Deterministic vs randomized where randomized one might create multiple or no leaders

In this paper: how economically can we solve implicit randomized leader election in terms of message complexity

Highlights of Paper
Time and message complexity of leader election related to

Algorithm that solves leader eletion for any graph using $\sqrt{n}$ messages and some rounds in terms of mixing time Also message lower bounds of $\sqrt{n}/\phi^{3/4}$ where ϕ is graph's conductance

Mixing time is bounded in terms of conductance by Sinclar et al. '89

Knowledge of the network size is critical for obtaining the message complexity upper bound

Implication: get sub-linear message and time complexity for well-connected graphs

For explicit leader election neem $\Omega(m)$ messages but for implicit can get away with fewer.

Note that implicit leader election + broadcast gives explicit leader election

Communication model is the CONGEST model

Algorithm:

1. Choosing contenders
2. Connect Contenders
3. Elect Leader

Random Walk
Important tool used in their algorithm Mixing time is time to find a random node in a random walk

Step 1
Thinning the crowd: choosing $\theta(\log n)$ contenders. Only a contender node can become a leader. Each node becomes a contender with probability $\theta(\log n/n)$

Step 2
Connect the contenders: find $\theta(\sqrt{n \log n})$ random nodes. Each contender sends $\theta(\sqrt{n \log n})$ random walks whose endpoints determnie the contender's target set. If mixing time is known: finding random nodes is easy; challenge is with unknown mixing time. Sufficient to satisfy "intersection" and "distinctness" property

Step 3
$\theta(\log n)$ contenders using their $\theta(\sqrt{n \log n})$ sized target sets collaborate to decide a leader

Theorem: can elect a unique leader in time proportional to mixing time and $\sqrt{n}$

Lower bounds
For lower bounds use graph conductance; a measure of graph bottlenecks

Cut conductance is the cross edges divided by the volume

Overall graph conductance is the min cut conductance across all cuts

Lower bound graph is a 4-regular random graph with $n^{1-\epsilon}$ nodes. Then replace each of thse nodes with cliques of size n^{ϵ}

Can show that the conductance of this graph is $1/n^{2\,\epsilon}$

Show lower bound by proof by contradiction. Suppose don't send that many messages. Then most messages are send inside a clique before it gets over the inter-clique edges. So many disjoint components that are nearly independent ; if they elect a leader you get more than one leader and if they don't elect a leader then no leader with constant probability

Conclusion is a lower bound in terms of $\sqrt{n}$ and $\phi^{3/4}$ messages (in expectation)

Criticality of Knowing n
Indistinguishibality argument and map LE to "bridge crossing"

Open Problems
Gap between upper and lower bounds Unknown neighbors vs known neighbors: what happens if nodes do know their neighbors?

2.4 Fabien Dufoulon on Beeping a Time-Optimal Leader Election (BA)

Give leader election algorithms for beeping model

Beeping model
Have communication graph and nodes with unique ideas. Time divided into synchronous rounds in which nodes can communicate or listen. When bepp transmit to all neighbors. Communication is very limited: messages have no content. Can only detect if 0 or more than 0 neighbors beeped

Very useful to have a leader

Want to solve explicit leader election as defined before

Lower bound of $\omega(D + \log n)$ and previous upper bound of $D \cdot \log n$.
Tightened gap with randomized leader election using k-balanced messages

Applications
MIS, multi-broadcast

2.5 Naoki Kitamura on Graph Exploration Using Constant-Size Memory and Storage (BA)

Model: an agent in an undirected simple graph

Agent begins exploration from initial node and visits all nodes and goes back to the initial node

Space complexity with poly(n) running time given. Their proposed algorithm is $O(1)$ memory space of agent and $O(1)$ storage space of each node. The running time of their algorithm is $O(nm)$. First algorithm with sublog memory and storyage and poly running time

Main idea: lex-DFS
If all neighbors visited backtrack else move to unvisited neighbor with smallest local port number Roughly: go back to the root node, trace in-stack nodes from the root

Application
Model can simulate TM with $O(n)$ bit tape

Future Work
Time comlexity lower bound Other algorithms in this setting

3 July 26, 2018

3.1 Keren Censor-Hillel on Barriers Due to Congestion and Two Ways to Deal with Them

“Lower bounds about problems that we care about a lot but cannot solve fast”

This talk is about:

1. Barriers due to congestion
2. Fast, good enough solutions (apx algorithms / testing algorithms)

Voltaire: ”Perfect is the enemy of the good” or literal translation ”better is the enemy of the good”

2 or 3 references to this quote in the context of apx algorithms

Highlights
-50 references
-270 London underground stations
-0 dances
-1 riddle

Planet CONGEST
Going to talk about other models as well to reason about CONGEST

London underground
#stations := n #passengers per minute := B Travel time?

This is just graph algorithms

Takes more time if more congestion

Don’t want nodes to just learn “everything”

Barriers
What takes time in this model?

1. Distances: if two nodes are at least 3 hops away it takes at least 3 rounds to transmit a message from one to another
2. Communication bottlenecks

Planet 2-Part Communication
To obtain LBs in CONGEST useful to reduce from 2-part communication
“Even models that are not formally the same model can allow us to reasons about one another”

Two parties; Alice and Bob each with k input bits, x and y; exchange bits with each other; try to compute some function $f(x, y)$ on input bits; complexity measure is total number of exchanged bits

E.g. set-disjointness: $f(x, y)$ is $\exists i$ such that $x_i = y_i$; lower bound of $\Omega(k)$ bits; this also holds for randomized algorithms

Simplest solution: Alice tells Bob enitre input, Bob computes $f(x, y)$, sends result back to Alice. Can’t do much better for set-disjointness

2-Party Communication -¿ CONGEST
Suppose A and B can build a graph $G_{x,y}$ such that some graph property P holds for $G_{x,y}$ iff $f(x, y) = 1$
They could build this graph and then try and determine if the property holds or not; one way they could do this is go to the CONGEST world and ask is there a good algorithm that answers this question and they

could simulate this algorithm; if two nodes on Bob's side of the graph want to send a a message to a node on A's side they could pass bits and anything within Bob's side can be simulated in Bob's mind
What's the cost of this simulation? Rounds · cut · B is $\Omega(\text{cost}(f(k)))$ and so rounds is $\Omega(\text{cost}(f(k))/\text{cut} \cdot B)$

While we can't control B we can try and control k and the cut size

<u>An example: Computing the Diameter D</u>
Do a reduction from 2-party communication
On A's side 2 cliques, all connected to a single node. B has the same thing. Then a perfect matching between all nodes. A diameter of 3.
But need the property (a diameter of 3 to depend on the inputs)
So augment graph with edges; put an edge between one of the n^2 possible inter clique edges if the value for this index is 0. Diameter could go down to 2.

Suppose they have a distributed algorithm that answers if diameter is 3. $k = \Theta(n^2)$, cut $= \Theta(n)$. Gives a lower bound of $\Omega(n/B)$: if I could compute diameter fast then I could solve set disjointness

<u>History</u>
A bunch of papers that use these sorts of lower bounds

<u>Perfect is the Enemy of the Good</u>
Computing the diameter D: -Perfect: $\Omega(n/B)$ for exact -Good: $O(D)$ for 2-approximation (BFS)

How well can we approximate? Do we need to pay a factor of 2?
Can get a 3/2-apx in $\tilde{O}(D + \sqrt{n})$
But slightly better: a $3/2 - \epsilon$-apx requires $\Omega(n/B \text{poly}\log(n))$; a sharp threshold at 3/2

Not known how threshold this threshold is: not known if you need $\sqrt{n}$ rounds for a 3/2-apx

<u>Computing a minimum vertex cover</u>
Can use 2-Part communication framework again
Same cliques on each side as before where input edges are according to input bits but not connected via a matching. Instead a "bit gadget". Namely $\log n$ bits to index vertices in the clique connected to its binary representation. A and B's bit gadgets connected in small cycles. Main thing is the cut is only logarithmic in the number of nodes. What happens is the size of the MVC is $4(k - 1 + \log k)$ iff the inputs are not disjoint; can't take 2 nodes out of a clique b/c then can't cover the inter clique edge. End up with a $\Omega(n^2/B \log n)$ lower bound.
Quadratic as above is the worst thing in the CONGEST model

<u>Better is the Enemy of Good</u>
Huge gap between perfect and good solutions for MVC so what about better solutions?
Can have fast 2 approximation

<u>Different Approach for Overcoming Lower Bounds not By APX Algorithms</u>
Via "subgraph detection"
Previous problems were "global". Subgraph detection problems are not global.

E.g. is there a 4-cycle in the graph?
Each side has $\Theta(n^{3/2})$ edges and no 4-cycles. Don't add all edges. (different from before because add an edge iff input bit is a 1 rather than a 0). Any 4 cycle cannot be within one side and so must have shape of 2 matching edges and edges on either side.

A solution: not an approximate solution; rather we can test for 4-cycle freeness; a solution that could be wrong some times

<u>Testing</u>
If G satisfies P output true and if G is ϵ-far from statisfying P you must output false
Far means even if you delete an ϵ fraction of the edges you still have the property

Moved to distributed framework in 2016

Many open probems in this area: how much better is better

Sometimes A-B is Insufficient
For weighted APSP above $\Omega(n)$
For 4-clique detection above $\Omega(\sqrt{n})$
For triangle detection: simply because for any triangle one of A and B know about the entire triangle (though we can for "triangle testing")

New technique for triangle detection: fooling views

3.2 Francois Le Gall on Sublinear-Time Quantum Computation of the Diameter in CONGEST Networks

Consider quantum version of CONGEST model; now are allowed to send quantum bits instead of usual bits; each node is a quantum processor
A quantum bit is one quantum particle; generalizes the concept of a bit

Background
Quantum Distributed Computing studied mostly in 2-party communication setting
Previous paper gives impossibility of quantum speedup for e.g. MST
Question: can quantum distributed computing be useful?
This talk: yes, can compute the diameter faster

Eccentrity of u is the max distance of u from another vertex

Eccentricity can be computed in $O(D)$ rounds but computing the diameter requires $\Theta(n)$ rounds, even for constant D

Main result: $O(\sqrt{nD})$ on round complexity of computing the diameter exactly; first gap between classical and quantum in distributed
Also a LB of $\tilde{\Omega}(\sqrt{n}D)$ which is unconditional and $\tilde{\Omega}(\sqrt{nD})$ which is conditional
Also a 3/2 approximation with even better round complexity

Now explain $O(\sqrt{nD})$

Focus on decision version; i.e. if $\exists u$ such that $ecc(u) \geq d$
Idea: use the technique "quantum search" (Grover's algorithm)

Grover's Algorithm
Given a function f as a black box. Want to find one $x \, inX$ such that $f(x) = 1$. In classical setting must do brute-force search over $x \in X$.
In quantum solving this problem can be done in $O(\sqrt{|X|})$ calls to the black box
E.g. application to quantum algorithm for Boolean SAT on M variables: X is all possible assigments and f is the Boolean formula; using quantum search takes $2^{M/2} \cdot \text{poly}(M)$; a quadratic speedup w.r.t. brute-force search

Distributed Quantum Search
Will apply this to diameter computation. $f : V \to \{0, 1\}$ is 1 iff $ecc(u) \geq d$. Will simulate Grover's centralized algorithm Network elects a leader and then leader will locally run this algorithm; everything can be locally simulated except for the calls to the blackbox f; any time you need a call to the black box you use the classical algorithm to compute the eccentricity.

$\sqrt{n}$ calls to the black box times D. With further work can put the D inside the square root.

Subtlteties
Quantum search is a quantum algorithm which works by manipulating *super positions*
Any classical algorithm can be converted into a quantum algorithm of similar compexity but allows inputs ”in superposition” so ok to convert classical algorithm into quantum algorithm; but this is for centralized so need to do this for distributed (key technical result)

Lower Bounds
Classical LB: Reduce DISJ to distributed computation of diameter
Unconditional: same reduction
Conditional: reduction with number of small messages then invoke Braverman LB

Open Problems
Try to apply these techniques to other distributed problems

3.3 Moti Medina on Property Testing of Planarity in the CONGEST Model

Motivation
Planarity detection: all nodes say yes iff an input graph is planar $\Omega(D)$ lower bound for detection: Take $k_{3,3}$ and replace each edge with a path of $\Theta(D)$ vertices Relaxing the problem can be useful: accept if planar; reject if “really” non-planar

Really non-planar just means that even after removing/adding ϵ fraction of edges still non-planar

Studied Problems
Subgraph-freeness Cycle-freeness Bipartiteness Planarity

Their Result
$\widetilde{O}(\log n)$ rounds w.h.p. Lower bound for testing H-minor-freeness: requires $\Omega(\log n)$ rounds

High Level Description

1. Planarity testing in each part
2. Partition so that low (strong) diameter and ϵm edges across nodes

Why is it ok to ignore the cut edges? Because if it’s epsilon far. . .

Terminology
Minor Forest decomposition Arboricity

Stage 1
Elkin forest decomposition -a forest decomposition of at most 3α forests where α is arboricity parameter Merging of parts -Not go into details

Requires a promise of bounded arboricity in each iteration; but we don’t know that for a fact; but if G has arboricity too large ”an evidence can be extracted”; so if it’s too large then just reject Invariant that we keep is diameter is inreased by a factor of 4 in each phase

Stage 2
Use planar embedding of Ghaf and Hae in each part. But still requires a promise that graph is planar; if this runs too long then evidence that graph is not planar; but if algo outputs an embedding then test if there are crossings of specific edges w.h.p.

Take home: can ϵ-test P with algorithms that require you promise P is satisfied

3.4 Krzysztof Nowicki on Congested Clique Algorithms for the Minimum Cut Problem

CONGESTED Clique Model
Same as CONGEST but fully connected

LBs as per Keren don't work because cut is always large; so we can hope for algorithms

In graph problems solved in this model the input graph is a subgraph of the communication graph; each player knows its set of incident edges and each player must know (some part) of the output

Min cut
Definition of a cut and the edges induced by a cut
Find a set A that induces as small a cut as possible

Results
$O(1)$ algorithm for min-cut apx. Also if min-cut is of size $\tilde{O}(n^{1/3})$ the algorithm is exact. The apx is additive $\tilde{O}(n^{-1/6})$ (focus of the talk)
Also $O(\log^3 n)$ round exact algorithm for min-cut (with memory-round tradeoff)
Also $O(\log^2 n)$ round exact algorithm

Ibaraki-Nagamochi Connectivity Certificates
Construction: sequentiall remove k spanning forests; build graph from original nodes and removed edges
Properties: if in original graph a cut of size smaller than k then it's present in the certificate; certificate is $\leq k$ connected iff G $\leq k$ connected
Naive Approach: time complexity proportional to the size of the min cut
Problem: spanning forset requires $\Theta(n)$ messages to describe

So instead studied random graph contractions by Karger (randomly contract edges)
Fact 1: contracting an edge that is not in min-cut preserves its size
Fact 2: A particular min-cut survives contraction to k vertices with probability at least $\Omega(k^2/n^2)$
Corollary: need $\Theta(n^2/k^2)$ trials for h.p.

Random Contracitons and Kruskal's MST Algorithm
Standard approach: random contractions by definition
Restated: permute the edges at random and contract the first eligible edge from a random permutation

Solve MST in parallel; as a result some set of players knows the MST; then just need to compute above certificates for component graphs

Parameters for Limited Bandwidth
Two stage aggregation

Karger's Sampling Technique
Apply Karger's sampling to sparser random subgraph

Summary
Start with input graph; run exact algorithm to verify if min cut is small; if it's not then test all probabilities where one probability gives graph that is neither too sparse nor too dense

3.5 Louis Esperet on Distributed coloring in sparse graphs with fewer colors

The Four Color Theorem
Is every planar graph 4-colorable?
Yes - Appel computer assisted proof (and the proof gives an $O(n^2)$ algorithm)
Much easier to do 5-coloring in $O(n)$ time

Natural question: can you go faster with a distributed (LOCAL) algorithm
Ideally would have something similar to the case of paths and trees (3 colorable quickly)

Goldberg et al. '88: Planar graphs can be 7-colored deterministically in $O(\log n)$ rounds
Proof: avg deg at most 6, linear num vertices with deg at most 6 by Markov so keep removing vertices for $O(\log n)$ rounds; if max degree is at most 6, 7-coloring is easy. List of available colors of remaining vertex is at least degree of a vertex +1, again this is easy

Gallai tree is a connected graph in which each connected component is a clique or an odd cycle
Borodin: whenever graph is not a Gallai tree and each vertex has a list of size at least its degree, can color

Now just remove vertices of deg at most 5 and remove vertices of deg 6 if they are in a non-Gallai component

Extending coloring from bad vertices to good vertices: compute a ruling set of good vertices, grow trees rooted at ruling set centers, color from leaves to root of trees, once arrive at root have to do more for centers of degree 6

Proof that we remove at least a linear number of vertices each time: if tree expands a lot then $\log n$ ball covers all nodes and so have low diameter, if not then many vertices of degree at least 7 so must also have many vertices of degree at most 5 (since avg degree is 6)

Main Theorem
If every subgraph of G has average degree at most $d \geq 3$ and G contains no K_{d+1} then can d-color G deterministically in $O(d^4 \log^3 n)$ LOCAL rounds.

Corollaries
Gives result for low arboricity
Can color planar graphs with 6 colors in $O(\log^3 n)$ rounds, triangle-free planar graphs with 4 colors

3.6 Yannic Maus on Improved Distributed Delta-Coloring

Main results
1: for all $\Delta > 3$, G not a clique, randomized Δ-coloring in $O(\log \Delta) + 2^{O(\sqrt{\log \log n})}$ rounds 2: For constant degree graphs, $\Delta \geq 3$, G not a clique, randomized $O(\log \log n^2)$ rounds

Distributed Vertex Coloring
Typical goal is $\Delta + 1$ coloring (b/c sequential algorithm just iterates through nodes and picks ok matches)

All of the 4 main problems have a simple greedy algorithm in common and can be completed to a full solution without changing previous stuff

Existence: Brooks theorem: if neither a clique nor an odd cycle then a Δ-coloring exists

Standard Technique: Shattering

1. Find a partial solution, remaning graph has small components
2. Solve small components

This doesn't work for Δ-coloring since small parts might not be solvable so need to do something else

Use **degree choosable components(DCC)**: a 2-connected non-clique graph (like cycles with chords)

Existential result: Any DCC can be degree-list colored

Thus, if left with some DCC at the end there at least exists a possible coloring

Main Framework

1. Remove node set I: find a ruling set of B_0 of small DCCs
2. Remove node set II: find layers $B_1, B_2, \ldots$
3. Color remaining graph
4. Add node set II and color it
5. Add node set I and color it: DCCs are by definition colorable

Crucially: every node has at most 1 neighbor in one layer below

Observations
DCCs are part of the graph that have 'slack'
Every node that is close to a selected DCC gets colored in 4

Don't really add DCCs but add things with similar properties; i.e. every node has slack close by

Gives results recap

Open problem: close the remaining gap for constant degree graphs

3.7 Andrey Kupavskii on Lower Bounds for Searching Robots, some Faulty

Some history: first questions asked by optimization community
A line, start from 0, search for a point on the line done somehow efficiently
Start at 0, move at constant speed 1, find target at distance $|x|$ within time $\lambda|x|$
Competitive ratio is 9
Upper bound is easy: go to 1, -2, 4, -8, 16 etc.

Note: Without $|x| \geq 1$ no competitive ratio is possible

Application: robots are cheap but faulty; e.g. 3 robots, but 1 is faulty (i.e. doesn't send a message when it passes the goal) and you don't know which
Previous work gives an upper and lower bound for this with a gap

Related to k robots searching on m rays emanating from 0 where $k < m$

Searching -¿ covering
Assume we have m rays and f faulty robots. Then each point must be covered $f + 1$ times

Theorem
q num robots, k faulty then $\rho := q/k$ and best competitive ratio is $2\frac{\rho^\rho}{(\rho-1)^{\rho-1}} + 1$

3.8 Uri Meir on Distributed Uniformity Testing

Property Testing
Large object we wish to test for a property by just taking a small number of samples

Distributed Property Testing
All talk about graphs; in particular, the communication graph
Their focus: distribution testing
Namely, decide if a distribution is the uniform distribution
Typically takes $\Theta(\sqrt{n}/\epsilon^2)$

Motivation: divide the costly 'sampling' process

1. If uniform then w.p. $\geq 2/3$ all nodes output 1
2. If ϵ-far then w.o. $\geq 2/3$ some node outputs 0

Complexity measure:

1. low communication
2. Small num of samples per node

Simple observations
Just have one node solve: good for communication, but expensive Cheap in samples: possibly bad for communication

Their Strategy
Divide into disjoint groups Gather samples within a group Each group outputs 1/0

But only a weak signal

Divide and Conquer
Want a few important nodes to have a strong enough signal

Results
For k important nodes: how many samples needed for each node? Standard rule Threshold rule: reject if T rejected Pick $T = \Theta(1/\epsilon^4)$

Technique: Uniformity Testing
Uniform distribution minimizes the probability of a collision
Far from uniform, far from minimal
Strategy: estimate collision probability by counting collisions

Algorithm
Collect samples into groups of $\Theta(\frac{\sqrt{n}}{\epsilon^2} \cdot f(k))$ samples Expect to rarely see acollision Only choice: each group accepts if it sees no collisions

For $f(k) = \frac{1}{\sqrt{k}}$ the signals are too weak

3.9 Goran Zuzic on Minor Excluded Network Families Admit Fast Distributed Algorithms

Suppose want to solve MST/min-cut on a distributed network

MST and min-cut can be solved on the family of networks without a fixed minor H in $\tilde{O}(D^2)$ rounds

Minor families
Planar, genus bounded, series-parallel, treewidth-bounded Excludes e.g. dense graphs

MST History
$\tilde{\Theta}(D + \sqrt{n})$ for general
$\tilde{O}(D)$ for planar, genus, treewidth
$\tilde{O}(D^2)$ on minor-free today
Latter two use 'shortcut framework'

Shortcuts
Framework to solve part-wise aggregation Definition of part-wise aggregation

Solve part-wise aggregation in R rounds $\rightarrow$ solve MST and min-cut in $\tilde{O}(R)$ time

Can't just propagate information within parts to get stuff to work because diameter might be very large

Definition of shortcut: assignment of edges to parts; also defined dilation and congestion

Goal
Prove good algorithms in minor-closed graph
Main tool: Graph Structure Theorem of Robertson-Seymour
A family of graphs without a minor H can be constructed as an $O(1)$-clique-sum of $O(1)$-genus graphs with $O(1)$ apices and vortices

Technical overview
Could take out apex of genus-bounded graph then take shortcut in resulting graph. Argue that when adding any "building block" preserve the quality of the shortcut

3.10 Przemek Uznanski on Population Protocols Are Fast (BA)

Defined population protocol model

Result
$O(1)$ states, $O(\text{poly}\log n)$ time randomized to compute basically whatever you want $O(1)$ states and $O(n^{\epsilon})$ time deterministically

Construction
Uses "phase clocks" to count time locally and get a feeling of the global time of the system

3.11 Janna Burman on Space-Optimal Naming in Population Protocols (BA)

Defined population protocols
Make fairness assumptions: weak says each pair interact infinitely often, global: inf often reachable configuration is reached infinitely often

Naming Problem
Start with anon population and want every agent to have a fixed and unique name

Measure state space complexity

3.12 Andréa W. Richa on A Local Stochastic Algorithm for Separation in Heterogeneous Self-Organizing Particle Systems (BA)

Interested in self-organizing collective systems e.g. water and oil

Particle system is trying to minimize the perimeter

Studied before but now have *heterogeneous* systems of particles
Model is asynchronous, anonmyous, no global info