Distributed algorithms and graph theory

Andrzej Czygrinow
Department of Mathematics and Statistics
Arizona State University

March 2, 2004
Outline of the talk

- Distributed networks.
- Vertex coloring (cycles).
- Edge coloring.
- Approximating the maximum matching.
Models of computations in CS

- Sequential RAM model.

- Parallel shared memory model.

- Distributed model.
Distributed network

1. Undirected graph:
   - Vertices - processors
   - Edges - communication links

2. We assume that there is a global clock and computations proceed in rounds.

3. In a round each vertex can:
   - Send information to its neighbors.
   - Receive information from its neighbors.
   - Perform some computations.
**Goal:** Estimate a given global function of the distributed network (say an independent set of vertices).

**Restrictions:** We must compute this function "fast". Number of rounds = poly-logarithmic in $n$.
An algorithm should be deterministic.
Problems:

- Function may not satisfy the required property.

- Not clear how to fix it: symmetry breaking.
**Obvious:** If the diameter of a graph is $K$ then we can do the computations in $O(K)$ rounds.

In particular if diameter is $O(\log n)$ then there is an efficient solution.
**Vertex Coloring Problem.** Let $G = (V, E)$ be a graph. Find the least $k$ and function $f : V \to \{1, \ldots, k\}$ such that if $vw \in E$ then $f(u) \neq f(v)$.

"Easy" problem: Color a cycle.
**Theorem 1 (Linial 92)** Any distributed algorithm that colors a cycle on $n$ vertices with 3 colors requires $\Omega(\log^* n)$ rounds.

**Theorem 2 (Cole-Vishkin 86)** There is a distributed algorithm which colors a cycle on $n$ vertices with 3 colors and runs in $O(\log^* n)$ rounds.

**Theorem 3 (Linial 92)** Any distributed algorithm that colors a cycle on $2n$ vertices with 2 colors requires $\Omega(n)$ rounds.

In general:

**Theorem 4 (Linial 92)** There is a distributed algorithm which colors a graph on $n$ vertices with $O(\Delta^2)$ colors and runs in $O(\log^* n)$ rounds.
Idea of the proof of Theorem 1

1. After $t$ steps of the execution of a distributed algorithm a vertex $x$ can know data from $2t$ vertices within distance $t$ of $x$. Consider $(x_1, \ldots, x_{2t+1})$.

2. Algorithm that runs in $t$ steps is therefore a coloring of these vectors with three colors.

3. Consider graph $B_{t,n}$ with vertices $=$ vectors of length $2t+1$ as above and two vectors $X$ and $Y$ connected if $X = (x_1, \ldots, x_{2t}, x)$ and $Y = (y, x_1, \ldots, x_{2t})$.

4. Distributed algorithm is a coloring of this graph with three colors... which is proper!
5. If $X = (x_1, \ldots, x_{2t}, x)$ and $Y = (y, x_1, \ldots, x_{2t})$ receive the same color then the coloring of the cycle fails if

$$y, x_1, \ldots, x_{2t}, x$$

is a segment in the cycle.

6. But $\chi(B_{t,n}) = \Omega(\log^{(2t)} n)$. 
Idea of the proof of Theorem 2

1. Start with a proper coloring and in each iteration reduce the number of colors.

2. To start, a vertex $v$ colors itself with $ID(v)$.

3. In each iteration, vertex $v$ gathers colors of all its neighbors $u_1, u_2$ and writes them in binary expansions.

4. Let $d(u_i, v)$ be the position of the first bit (from left to right) on which the colors of $u_i$ and $v$ differ. Let $b_v(u_i)$ be the value of the bit in the color of $v$.

5. Vertex $v$ colors itself with a vector

$$[(d(u_1, v), b_v(u_1)), (d(u_2, v), b_v(u_2))].$$
1. How many bits a vertex needs to write its new color?

\[ 2(1 + \log(BITS)) \]

where \( BITS \) is the number of bits used to write a color in the previous iteration.

**The number of colors is rapidly shrinking!**

2. The coloring is proper. Take vectors of \( u \) and \( v \). If there is an \( i \) such that \( d_i(v) \neq d_i(u) \) then two vectors are different. Otherwise look at the position \( j \) such that \( u_j = u \). Then \( d_j(v) = d_j(u) \) (Distances are the same).

But \( b_j(v) \neq b_j(u) \) (Definition of \( d(u_j, v) \)).
Edge coloring  Let $G = (V, E)$ be a graph. Find a number $L$ and function $f : E \rightarrow \{1, \ldots, L\}$ such that edges that share an endpoint receive different colors.

Chromatic index of $G$, $\chi'(G)$, is the least such $L$.

Theorem 5 (Vizing) \hspace{1cm} \Delta(G) \leq \chi'(G) \leq \Delta(G) + 1.$

Problem 1 (Panconesi) Does there exist an efficient distributed algorithm which colors a graph $G$ with $O(\Delta(G))$ colors?
Theorem 6 (C., Hanckowiak, Karonski) There is a distributed algorithm which in $O(\log^4 n)$ rounds finds a proper edge-coloring with $O(\Delta \log n)$ colors.

A bipartite graph $H = (A, B, E)$ is called a $D$-block if for every vertex $a \in A$, 
\[ \frac{D}{2} < \deg_H(a) \leq D. \]

Definition 1 An $(\alpha, \beta)$-spanner of a $D$-block $H = (A, B, E)$ is a sub-graph $S = (A', B, E')$ of $H$ such that the following conditions are satisfied.

1. $|A'| \geq \alpha |A|$.

2. For every vertex $a \in A'$, $\deg_S(a) = 1$.

3. For every vertex $b \in B$, $\deg_S(b) < \frac{\beta}{D} \deg_H(b) + 1$. 
**Theorem 7** Let $H = (A, B, E)$ be a $D$-block. There is a distributed algorithm which finds in $O(\log^3 n)$ rounds a family of $O(D)$ disjoint, $(\frac{1}{2}, 16)$-spanners of $H$.

**Idea:** Color edges of the graph using two colors 0,1 so that for each $v$ the degree of $v$ in the monochromatic graph is approximately $\deg(v)/2$. Process is repeated $O(\log D)$ times. This sequence of bits (for example 0010101) determines one spanner from the family.
How to color a block?

Three observations:

1. Each spanner can color itself using $O(\Delta/D)$ colors.

2. Family can be colored using $O(\Delta)$ colors.

3. Each spanner contains $|A|/2$ vertices from $A$. The number of edges in the family is at least $c|E|$ for some constant $c$. 
How to color a bipartite graph?

Given: bipartite graph \((A, B, E)\).

Method:
1. Split a graph into \(O(\log \Delta)\) blocks. The \(i\)th block, \(A_i\) = vertices with degree in \((\Delta/2^i, \Delta/2^{i-1}]\).
2. Compute families of spanners in each block (in parallel) and color them

3. For every "large" star if there are edges in the star with the same color, then un-color all but the one that touches the block with the largest value of $D$. 
Let $E_i$ be the number of edges in the $i$th block. Then in each iteration

$$\sum_i |E_i| D_i$$

decreases by a constant fraction.
An edge from the \( i \)th block has "weight" \( D_i \) in the formula above, but the sum of weights of edges from blocks 2, 3, 4, \ldots is

\[
\Delta \sum_{i \geq 1} \frac{1}{2^i} < \Delta
\]

On the other hand, we keep the weight of at least \( \Delta/2 \).

We "keep" at least 1/3 of the value of the sum.
How to color a graph?
1. Give orientation to edges.
2. Split each vertex into two.
3. Color the bipartite graph.
4. Glue the vertices together. One can get monochromatic cycles or paths. Find a large matching in each monochromatic component.
Matchings
A matching is a sub-set of edges $M$ such that no two edges from $M$ share an endpoint.

**Maximal matching** - matching $M$ such that there is no $M'$ such that $M$ is a proper subset of $M'$.

**Maximum matching** - matching $M$ such that $|M|$ is the largest possible.

**Theorem 8 (Hanckowiak, Karonski, Panconesi)**
There is a distributed algorithm which finds a maximal matching in $O(\log^4 n)$ rounds.

**Observation:** If $M$ is maximal, $M^*$ maximum then $|M| \geq \frac{1}{2} |M^*|$.  

**Problem:** Can you find a larger matching $M$?
Theorem 9 (C., Hanckowiak, Szymanska) There is a distributed algorithm which finds a matching $M$ such that
\[ |M| \geq \frac{2}{3}|M^*|. \]
The algorithm runs in $O(\log^6 n)$ rounds.

Theorem 10 (C., Hanckowiak) Let $G$ be a graph without cycles of lengths $3, 5, \ldots, 2k-1$. Then there is a distributed algorithm which finds a matching $M$ such that
\[ |M| \geq \frac{k}{k+1}|M^*|. \]
The algorithm runs in $O(\log^D n)$ rounds, where $D = D(k)$. 
Idea of the algorithm

• Find a maximal matching $M$ using Theorem 8.

• Find a maximal set of $M$-augmenting paths.

• Augment the paths.
Finding paths requires work

- Reduce graph $G$ and matching $M$ to a special layered graph.

- Find maximal set of paths in the layered graph.

- Translate the paths back to $G$. 
Reduction

G

Layered graph
Paths in the layered graph

- Iterate over blocks.
- In a block find a spanner.
- Create an auxiliary multi-graph.
- Call the same algorithm recursively.
- In other words: somewhat complicated.
Translation

- Paths in the layered graph correspond to paths in graph $G$ (there are no odd cycles)

- ... but paths do not need to be disjoint.

- Consider the graph of paths.

- Find a "maximal weighted independent set" in the graph of paths.
Problems and future work

• Edge coloring using $O(\Delta)$ colors.

• Approximating the maximum matching in arbitrary graphs.

• Faster algorithms.

• **Problem:** Maximal Independent Set.