Easy graph algorithms under hard constraints

GraphEx Symposium 2016

Paul Burkhardt

U.S. National Security Agency
Research Directorate

May 19, 2016

Graphs are everywhere!

Why graph?

Graph-theoretic approaches are appealing to many fields

- simple data abstraction
  - binary relationships — nodes and links
- powerful algorithmic approaches

The expanding field of graph research

Results from Google Scholar

The number of articles containing "graph", "algorithm" and "vertex" or "edge" has more than *tripled* in the last ten years!

576,000 articles through 2006

2,020,000 articles through today

The expanding field of graph research

Results from Google Scholar

The number of articles containing "graph", "algorithm" and "vertex" or "edge" has more than *tripled* in the last ten years!

576,000 articles through 2006

2,020,000 articles through today

Google Scholar results for the following terms added to first query

Up to four-fold increase across non-traditional disciplines

neuroscience 4× increase: 227 → 1,020

biology 4× increase: 2,000 → 7,720

chemistry 3× increase: 1570 → 4,580

Graphs in Biology

Pathways in Protein-Protein Interaction (PPI) networks

Finding signaling pathways in yeast is analogous to finding k-length simple paths in graphs [6]

- simple path has no repeated vertices

Comparing PPI networks

Comparing yeast and fly protein interaction networks [8]

- find the maximum common subgraph (subgraph isomorphism)

Graphs in Chemistry

Chemical reactions

Identify routes in chemical reactions [9]

- cyclic subgraphs — route of reaction

Dissociation pathways

Lowest-cost pathway for six-dimensional free energy surface of 1 million grid points [10]

- Dijkstra's shortest-path

N-body interactions

Enumerate n-body intramolecular interactions using line graphs [5]

Graphs in Neuroscience

Conscious awareness

Awareness is linked to the brain's functional connectivity [2]

- decreased modularity (functional segregation)
- increase in connectivity across modules, i.e. functional regions

Graphs in Neuroscience

Conscious awareness

Awareness is linked to the brain's functional connectivity [2]

- decreased modularity (functional segregation)
- increase in connectivity across modules, i.e. functional regions

Schizophrenia

Schizophrenia brain is less "small world" and more "random graph" [4, 1]

- decreased modularity (functional segregation)
- increase in connectivity across modules, i.e. functional regions
- lower probability of hubs (high degree centrality)

Graphs in Neuroscience

Conscious awareness

Awareness is linked to the brain's functional connectivity [2]

- decreased modularity (functional segregation) ✓
- increase in connectivity across modules, i.e. functional regions ✓

Schizophrenia

Schizophrenia brain is less "small world" and more "random graph" [4, 1]

- decreased modularity (functional segregation) ✓
- increase in connectivity across modules, i.e. functional regions ✓
- lower probability of hubs (high degree centrality)

Graphs in Neuroscience

Conscious awareness

Awareness is linked to the brain's functional connectivity [2]

- decreased modularity (functional segregation) ✓
- increase in connectivity across modules, i.e. functional regions ✓

Schizophrenia

Schizophrenia brain is less "small world" and more "random graph" [4, 1]

- decreased modularity (functional segregation) ✓
- increase in connectivity across modules, i.e. functional regions ✓
- lower probability of hubs (high degree centrality)

Greater self-awareness in Schizophrenia?

Graphs in Neuroscience

Alzheimer's disease (AD)

Alzheimer's disease (AD) exhibit aberrant brain network [3]

- longer shortest-paths (less effective information transfer)
- largest connected-component is smaller in AD patients

Technology shift

Google Scholar results for the following terms added to first query

gpu 13× increase: 1,990 → 25,600

database 6× increase: 78,100 → 497,000

rdf 5× increase: 2,500 → 13,800

Technology shift

Google Scholar results for the following terms added to first query

gpu 13× increase: 1,990 → 25,600

database 6× increase: 78,100 → 497,000

rdf 5× increase: 2,500 → 13,800

Graphs on GPUs

Graph algorithms expressed in the language of linear algebra...

- sparse matrix products are natural for GPU acceleration

Technology shift

Google Scholar results for the following terms added to first query

gpu 13× increase: 1,990 → 25,600

database 6× increase: 78,100 → 497,000

rdf 5× increase: 2,500 → 13,800

Graphs on GPUs

Graph algorithms expressed in the language of linear algebra...

- sparse matrix products are natural for GPU acceleration

Graph DB

Graphs are becoming the new relational model!

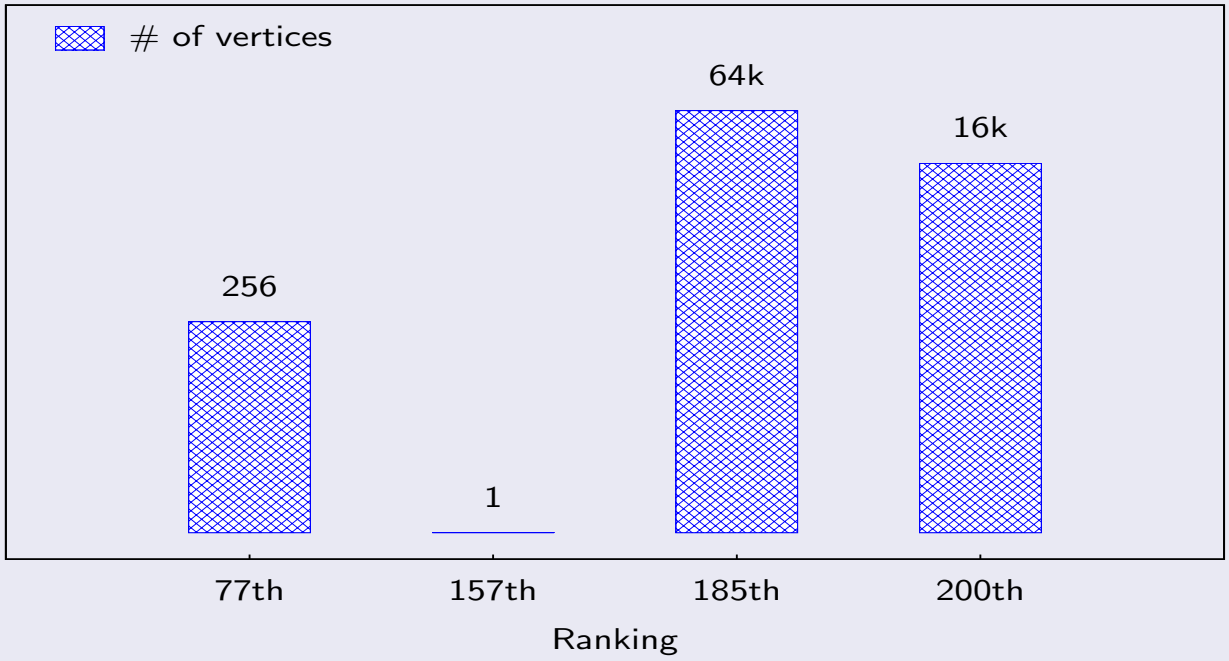
- graph databases
- semantic graphs

Who needs Big Graphs?

Graph500 scale ≤ 16

If only 64k vertices was more than anyone needed...

Source: Graph500 November 2015 List

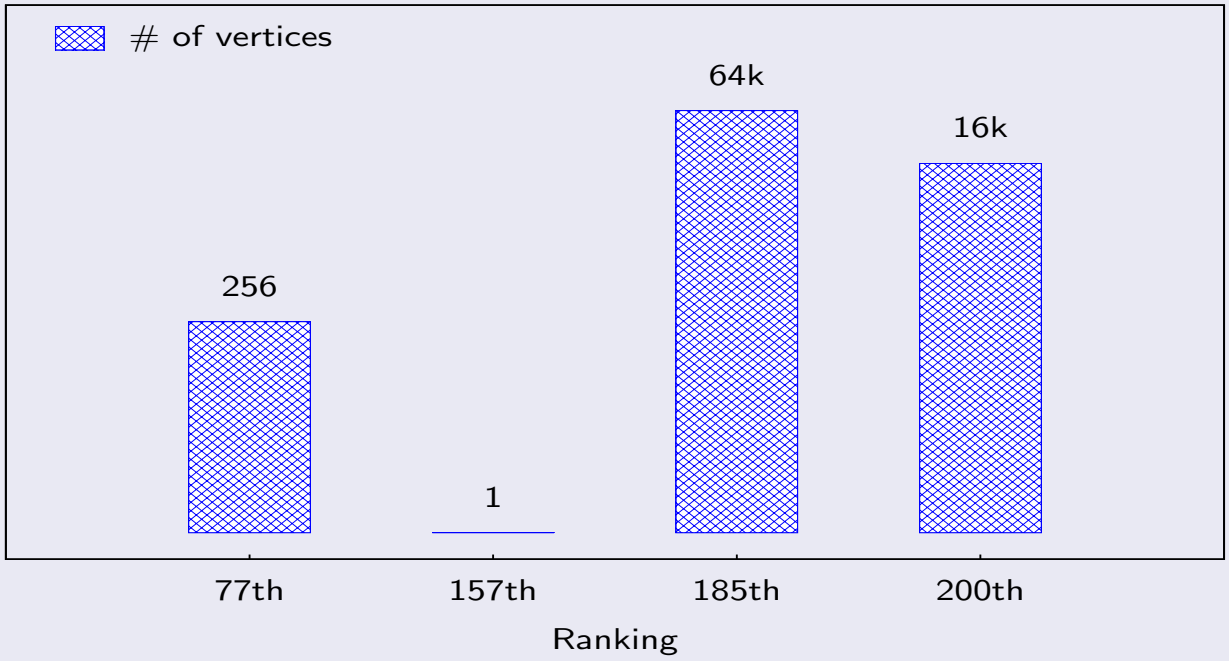


Who needs Big Graphs?

Graph500 scale ≤ 16

If only 64k vertices was more than anyone needed...

Source: Graph500 November 2015 List



The 77th ranked competitor can assign one thread to each vertex!

- Intel Xeon E5-2670 — 128 cores
- 256 threads with simultaneous multithreading (SMT)

More focus on increasing graph sizes

The challenge is with Big Graphs

Increasing data and problem sizes lead to bigger graphs. . .

- everything gets harder with increasing scale

Google Scholar results for the following terms added to first query

distributed 60× increase: 17,400 → 1,150,000

parallel 14× increase: 74,100 → 1,030,000

scalable 7× increase: 19,100 → 134,000

The need for graph analytics!

Trending

The increase in graph analysis across different domains can be due in part to

- simple and compact representation of complex data
- practical but powerful algorithms — many of which are easy to implement

Data as a graph

A graph is simple but not all data looks like a graph...

- some datasets are a natural fit, e.g. human brain
- other data has to be mapped...

Graph algorithms need a graph

Is your data a graph?

- Your data may not appear as a network. . .

Graph algorithms need a graph

Is your data a graph?

- Your data may not appear as a network...
- But any pairwise collection of data points is a graph!

Graph algorithms need a graph

Is your data a graph?

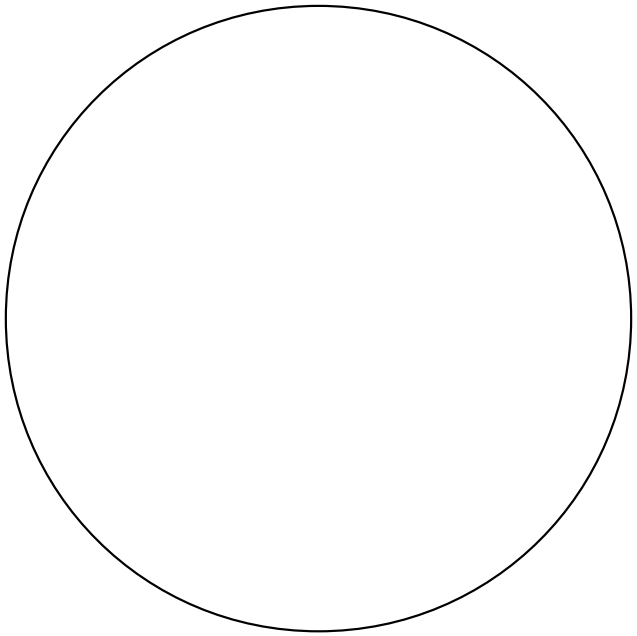
- Your data may not appear as a network...
- But any pairwise collection of data points is a graph!

Let's play a game!

Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

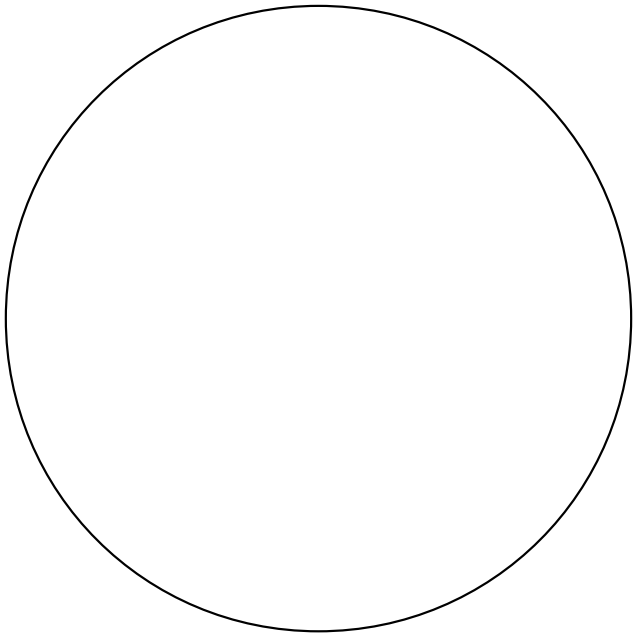


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard

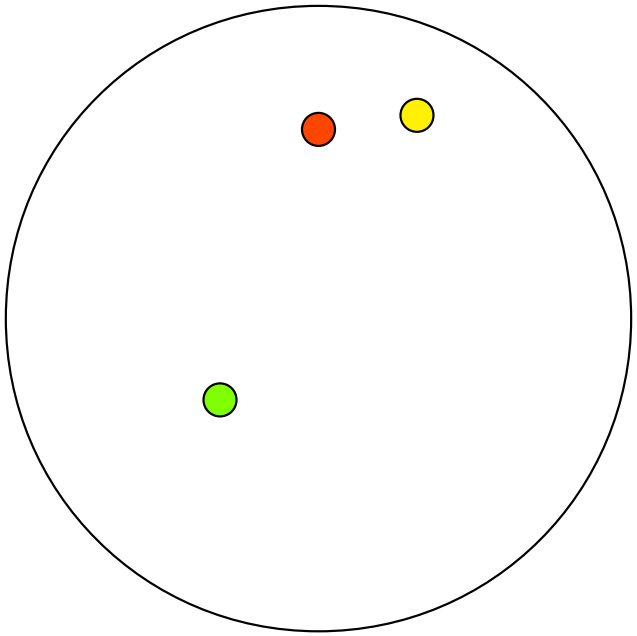


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard

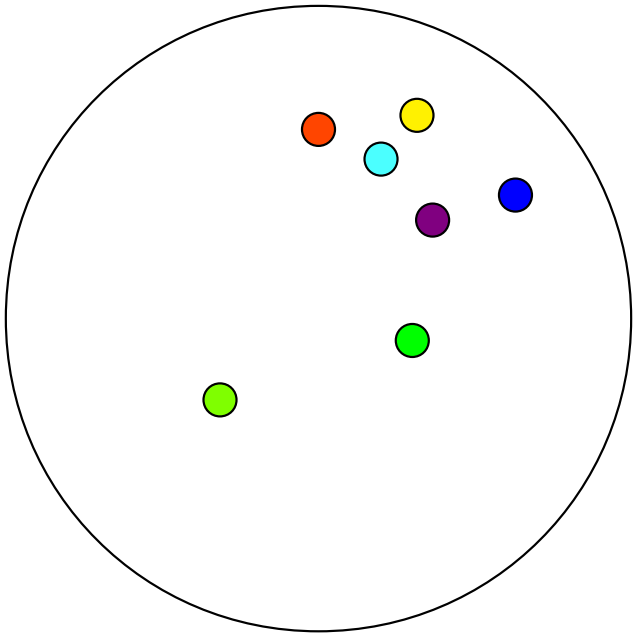


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard

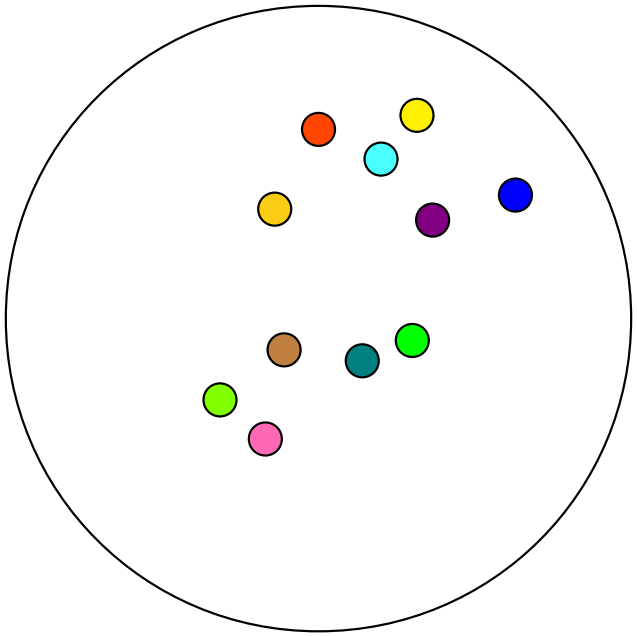


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard

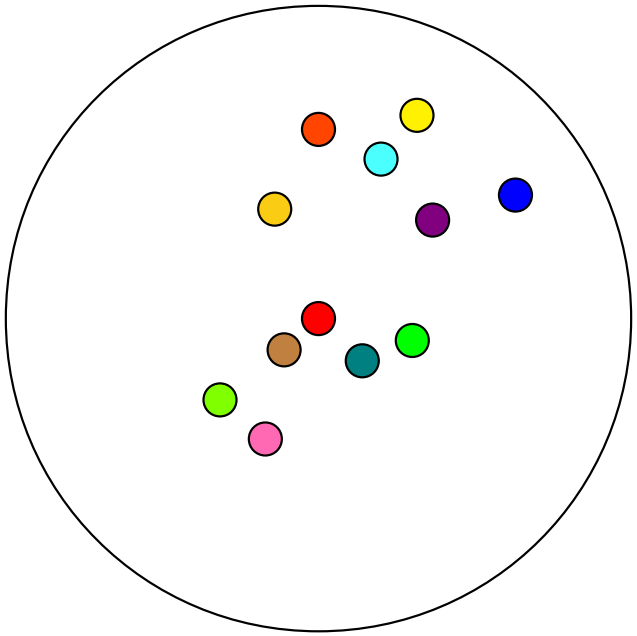


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard

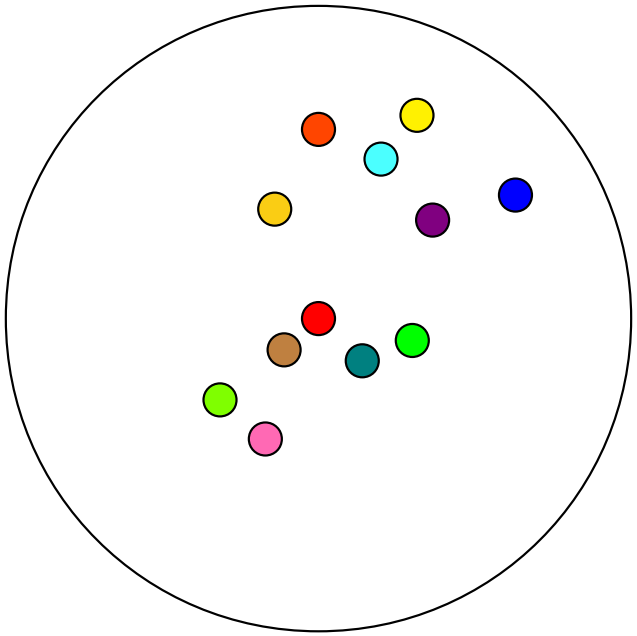


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard
2. draw lines to split dartboard into quadrants

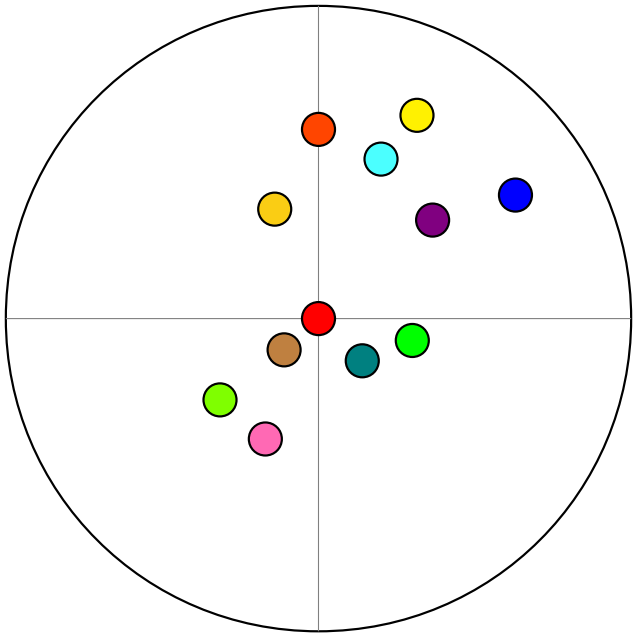


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard
2. draw lines to split dartboard into quadrants

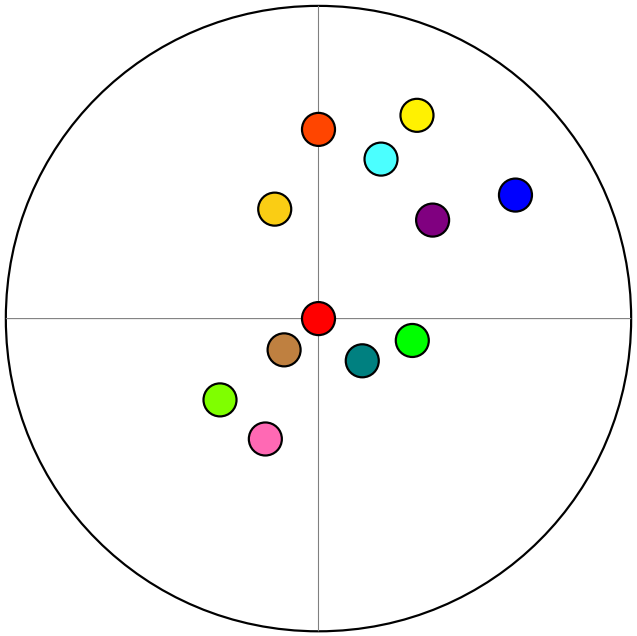


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard
2. draw lines to split dartboard into quadrants
3. all darts in the same quadrant are neighbors; they form a clique

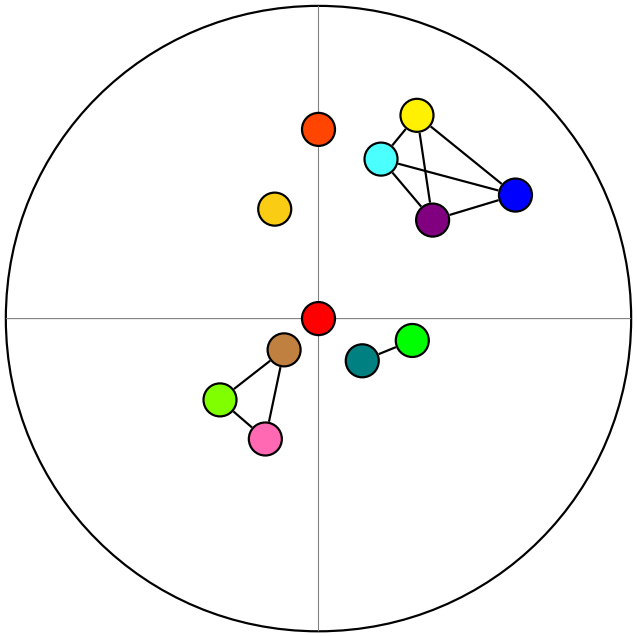


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard
2. draw lines to split dartboard into quadrants
3. all darts in the same quadrant are neighbors; they form a clique

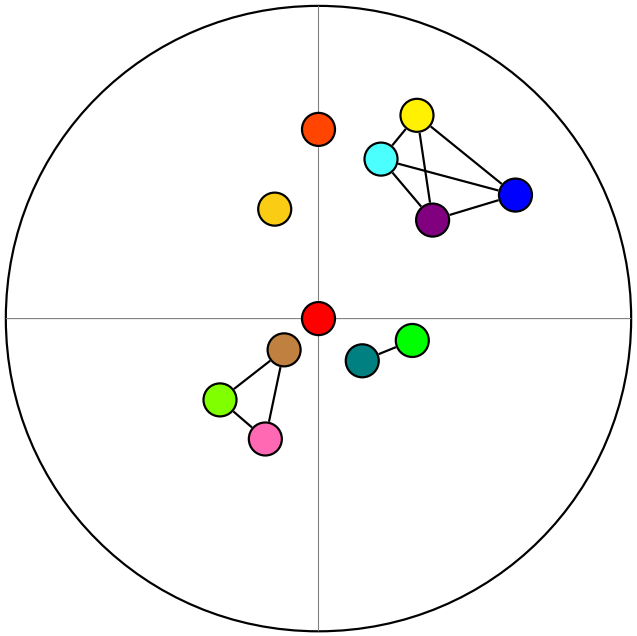


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard
2. draw lines to split dartboard into quadrants
3. all darts in the same quadrant are neighbors; they form a clique
4. a dart on the line separating two quadrants is a neighbor of the nearest dart in each quadrant

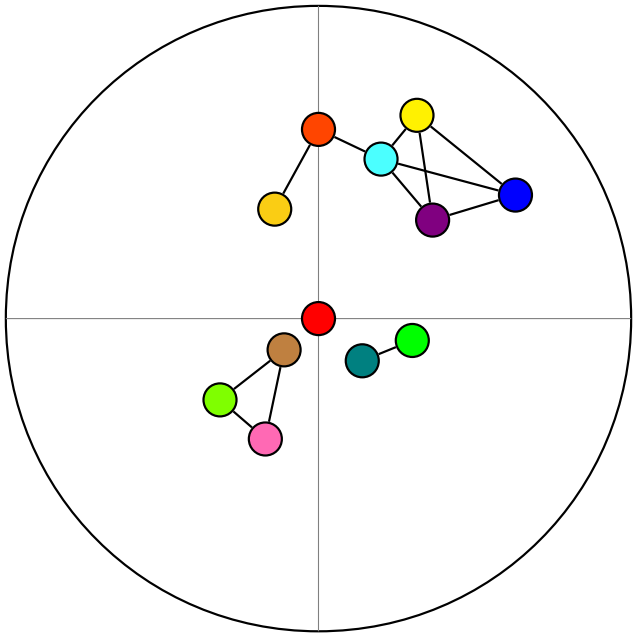


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard
2. draw lines to split dartboard into quadrants
3. all darts in the same quadrant are neighbors; they form a clique
4. a dart on the line separating two quadrants is a neighbor of the nearest dart in each quadrant

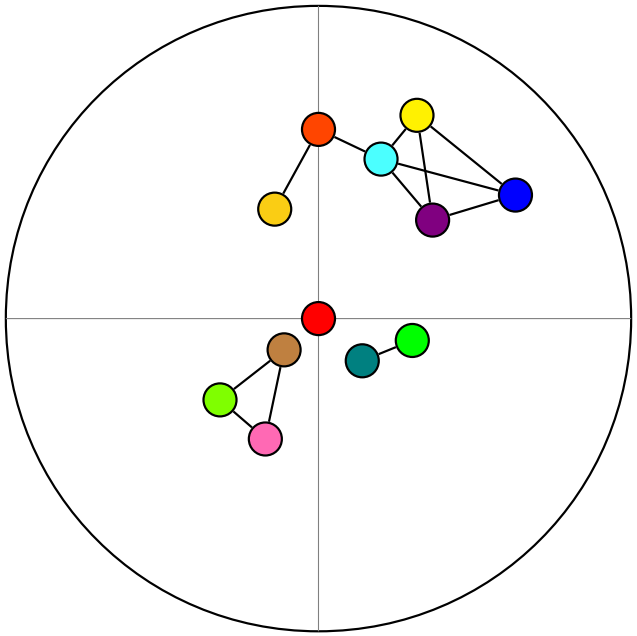


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard
2. draw lines to split dartboard into quadrants
3. all darts in the same quadrant are neighbors; they form a clique
4. a dart on the line separating two quadrants is a neighbor of the nearest dart in each quadrant
5. a dart in the bulls-eye is a neighbor to the nearest dart in all quadrants

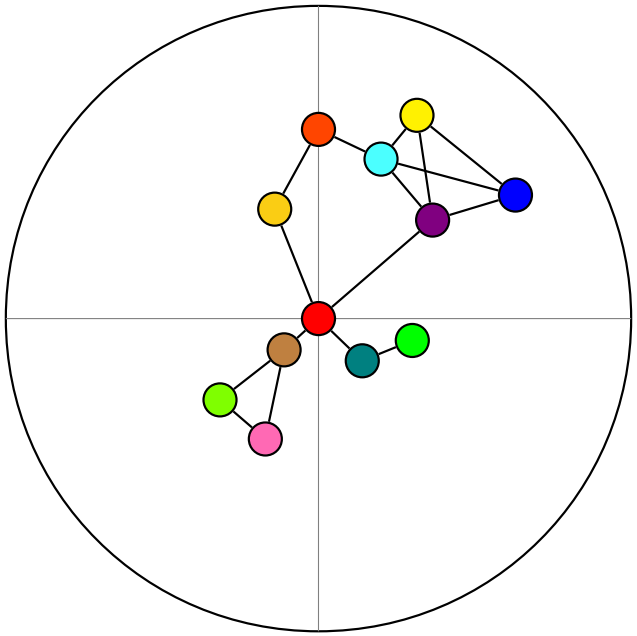


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard
2. draw lines to split dartboard into quadrants
3. all darts in the same quadrant are neighbors; they form a clique
4. a dart on the line separating two quadrants is a neighbor of the nearest dart in each quadrant
5. a dart in the bulls-eye is a neighbor to the nearest dart in all quadrants

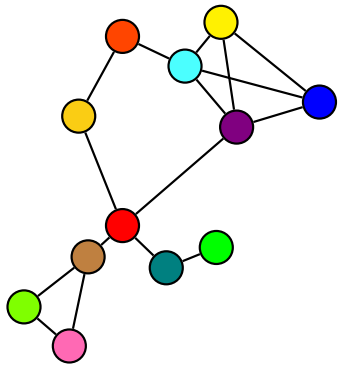


Graphing the Game of Darts

Example

Go to nearest pub and find a dartboard...

1. throw darts at a dartboard
2. draw lines to split dartboard into quadrants
3. all darts in the same quadrant are neighbors; they form a clique
4. a dart on the line separating two quadrants is a neighbor of the nearest dart in each quadrant
5. a dart in the bulls-eye is a neighbor to the nearest dart in all quadrants

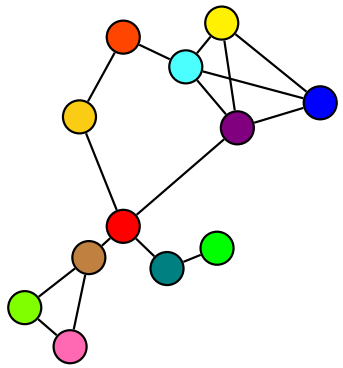


Dart Graph

Simple graph generator

Our game was a simple graph generator. . . not unlike other generators

- we created some recognizable structures. . .
- cycles, cliques, cores, components, clusters. . .



Now analyze the graph

Graph analysis

What can we learn from simple graph analysis?

Now analyze the graph

Graph analysis

What can we learn from simple graph analysis?

- I'm not very good at darts?

Now analyze the graph

Graph analysis

What can we learn from simple graph analysis?

- I'm not very good at darts?
- Much more – we can identify structure and cohesiveness

Now analyze the graph

Graph analysis

What can we learn from simple graph analysis?

- I'm not very good at darts?
- Much more – we can identify structure and cohesiveness
  - connected components

Now analyze the graph

Graph analysis

What can we learn from simple graph analysis?

- I'm not very good at darts?
- Much more – we can identify structure and cohesiveness
  - connected components
  - cores

Now analyze the graph

Graph analysis

What can we learn from simple graph analysis?

- I'm not very good at darts?
- Much more – we can identify structure and cohesiveness
  - connected components
  - cores

But we'll do this at a larger scale and under hard constraints. . .

Now analyze the graph

Graph analysis

What can we learn from simple graph analysis?

- I'm not very good at darts?
- Much more – we can identify structure and cohesiveness
  - connected components
  - cores

But we'll do this at a larger scale and under hard constraints. . .

New algorithms

We'll describe our easy algorithms under hard constraints

Graph data is hard. . . really hard!

Real-world graphs represent irregular data

- topology is irregular thus. . .
- difficult to leverage computer memory hierarchy leading to. . .
- increasing latency and bandwidth costs and. . .
- high degree vertices cause hot-spots

Hard constraints for graph algorithms

Big Data constraints

Constraints commonly encountered with Big Data.

- graph does not fit in memory
- share nothing (no globally-shared data)
- no random access

Harder constraints. . .

For good measure we'll add the following constraints

state-less	tasks cannot save local information between steps
constant-memory	tasks have fixed memory with respect to input
data-stream	tasks cannot re-read input within the same step

Hardest constraint of all!

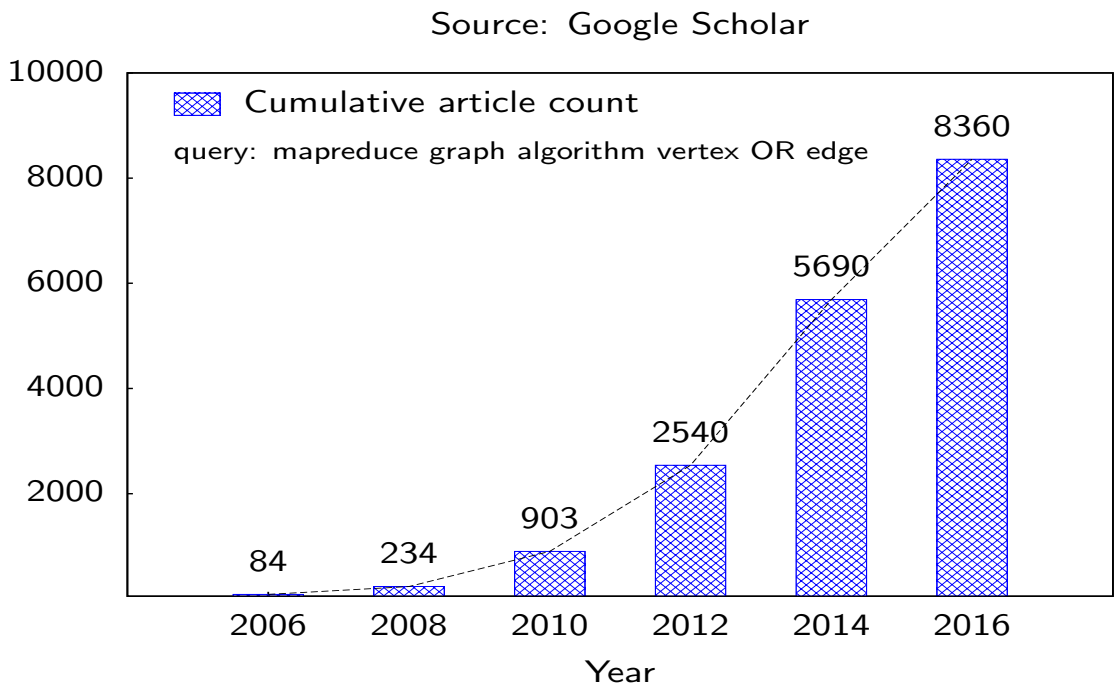
Hardest constraint of all!

We'll design the algorithms in MapReduce!

Results from Google Scholar

100-fold increase in MapReduce graph algorithm articles in the last ten years!

- spike is common in early phase of new technology. . .
- but still an active research area



Why MapReduce?

Benefits

The MapReduce computational model provides many benefits

- distributed computation
- compute migration
- overlap of communication with computation
- not memory-bound
- fault-tolerant. . .

Growing research interest

- improvements in theoretical model of computation
- increase in algorithm development

Exercising constraint

Limitations for a gain. . .

In this exercise the hard constraints will limit algorithm design choices but help to

- maximize scalability
- improve parallelism
- conserve compute resources
- support streaming computation

Terms and Definitions

For a simple, undirected graph $G = (V, E)$

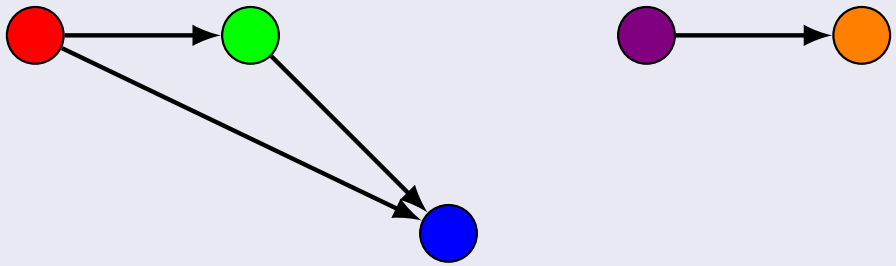
- with $n = |V|$ vertices and $m = |E|$ edges where ...
- $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix of G; $A^T = A$
- $N(v) = \{u \in V \mid (v, u) \in E\}$ is the neighborhood of v
- $d(v) = |N(v)|$ is the degree of v
- $d_{max} = max\{d(v) \mid v \in V\}$ is the maximum degree
- $d(u, v)$ is the distance, i.e. shortest-path, from u to v
- $D(G)$ is the diameter of G, i.e. longest shortest-path in G

Connected Components in graphs

Group by connectivity

Group vertices that are reachable by a path, albeit not necessarily mutually reachable.

How many components do you see?

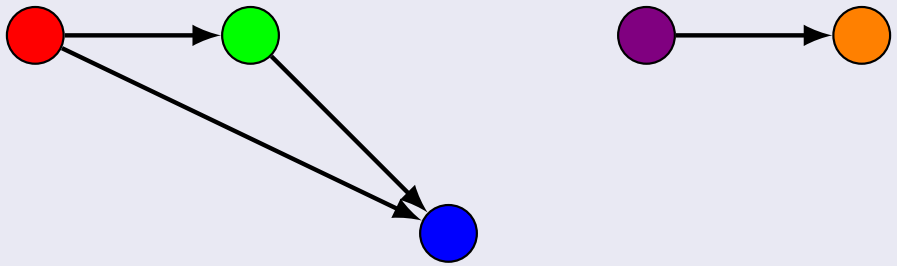


Connected Components in graphs

Group by connectivity

Group vertices that are reachable by a path, albeit not necessarily mutually reachable.

How many components do you see?

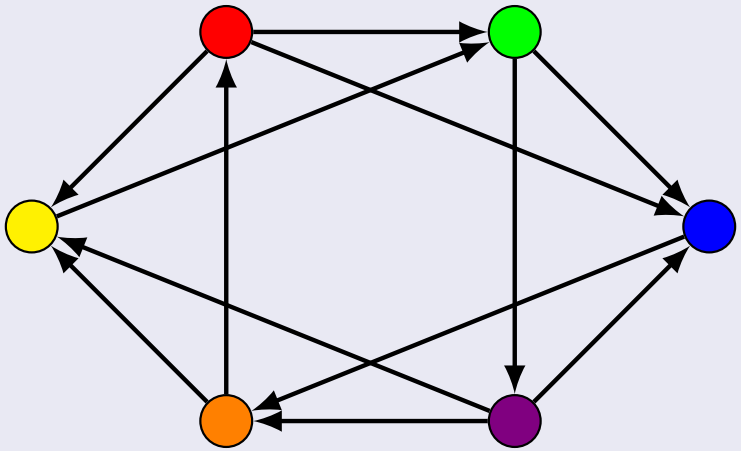


Is it 2 or 5?

Strongly Connected Component

Definition

A connected component in which a path exists between every pair of vertices.



Trivial path...

A singleton vertex is a connected component because it is trivially connected to itself; $d(v, v) = 0$

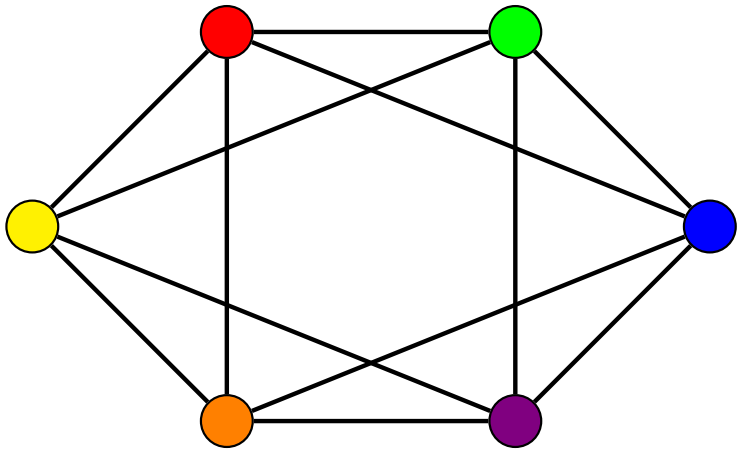
Connected Components in Undirected Graphs

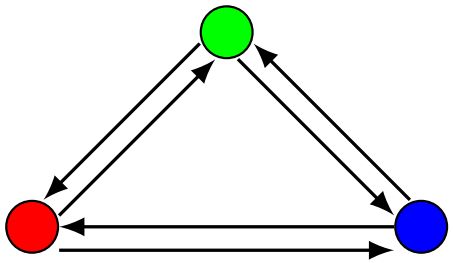
Strongly connected

Paths are symmetric in an undirected graph, thus a connected component in an undirected graph is strongly connected.

Undirected G. . .

The remainder of this presentation will focus on undirected graphs.





Easy Connected Components under hard constraints

Theorem 1

Given an edge list of G and a list L then the algorithm in Listing 1 takes $O(m \log^2 n)$ time.

Proof.

The edges are sorted so (v, w) is first where w is the minimum vertex ID for all endpoints of v. The time to sort all edges is

$$O(\sum_v d_v \log d_v) \leq O(m \log d_{max}) \leq O(m \log n)$$

There are $O(m)$ tests for each edge in L, and all other operations including inserts into L' take $O(1)$ time where L' is a temporary list. The time for each step is then $O(m + m \log n) \in O(m \log n)$. Every vertex v with a neighbor u, where $u \prec v$, will receive the minimum from $N(u)$. This contracts the graph by half after each step. Thus the overall time for $O(\log n)$ steps is $O(m \log^2 n)$! □

Listing 1: Label propagation Connected Components

```
L ← sorted E
count ← 1
while count > 0 do
    count ← 0
    create new list L'
    last ← ∞
    for all (v, u) ∈ L do
        if v ≠ last then
            w ← u
            last ← v
            if w < v then
                add (v, w), (w, v) to L'
            end if
        else if w < v then
            add (u, w) to L'
            count ← count + 1
        end if
    end for
    sort L'
    L ← L'
end while
```

Connected components under hard constraints. . .

Benefits

- does not re-read the entire graph at each step
- fast drop-off in output size
- neighborhood of a vertex is ignored if the vertex ID is less than the minimum of all neighbors. . .
- completes in $O(\log n)$ rounds

MapReduce Connected Components

Connected Components algorithm

- $N[v]$ is the sorted set of neighbors of v
- Constant, $O(1)$, working memory

Round $1..O(\log n)$

Map: Ordered vertex pair for secondary-sort

$$\langle v, u \rangle \longrightarrow \langle (v, u), u \rangle$$

Reduce: Values are sorted so the first value, w, is the minimum

$$\langle v, \{u \in N[v]\} \rangle \longrightarrow \langle v, w \rangle, \langle w, v \rangle, \{\langle u, w \rangle\} \iff w \prec v, w = \min N[v]$$

Final Round

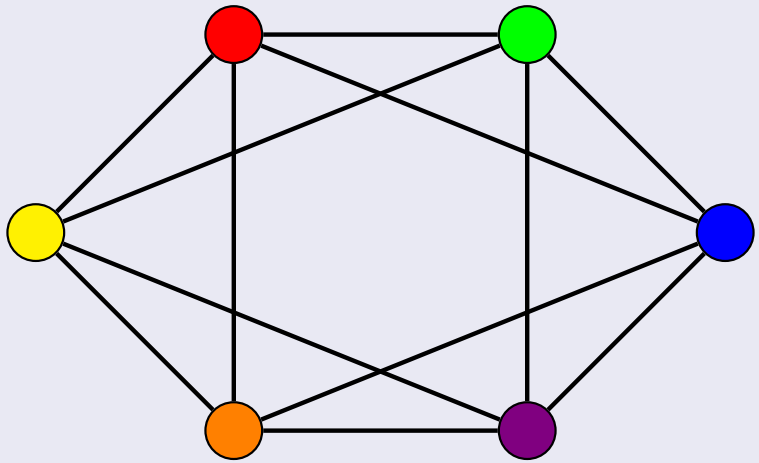
Map: Organize (v, u) pairs

$$\langle v, u \rangle \longrightarrow \langle v \prec u, u \rangle \iff v \prec u$$

Cores in Graphs

(loose) Definition

A core is a maximal subgraph in which vertices are well-connected



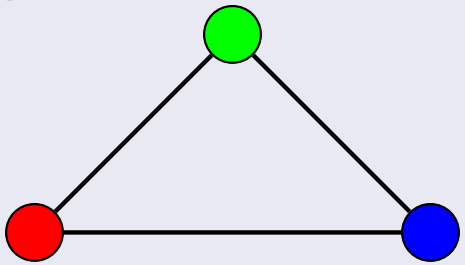
Cores are cohesive networks

Significant number of vertices must be removed for the core to become disconnected. . . cliques are maximally connected

Recall the definition of a clique...

Definition

A clique is a graph in which all pairs of vertices are connected, e.g. triangle



Completeness

A n-clique is a maximally complete graph of order n

- all degrees = n-1
- graph diameter = 1
- number of edges = n(n-1)/2

Cores are a relaxation of cliques

Cores are more loose then cliques...

Less strict in degree, connectivity, reachability, etc.

Types of cores

k-plex	each vertex is not connected to at most k-1 other vertices
k-club	diameter is at most k
γ-quasi-clique	density is at least γ
k-core	each vertex has degree at least k
k-truss	endpoints of each edge share at least k-2 neighbors

Reviewing the k-core

Introduced by Stephen B. Seidman in 1983 [7]

A k-core is a maximal connected subgraph with minimum degree k.

Hierarchy of k-cores

A k-core is a subset of the $[1..(k-1)]$-cores

$$\text{k-core} \subseteq \text{(k-1)-core} \subseteq \text{(k-2)-core} \ldots \subseteq \text{1-core}$$

Easy k-Core under hard constraints

Theorem 2

Given a sorted edge list of G, a list L, and $k \in O(\sqrt{m})$, then the k-core algorithm in Listing 2 is possible in $O(md_{max})$ time, and if d_{max} is constant the time is $O(m)$.

Proof.

At each step, all operations take $O(1)$ time including inserting an edge into the temporary list T_v and inserting T_v into the temporary list L'. Therefore the first pass takes $O(m)$ time, leaving at most $O(m/k)$ vertices remaining with $O(\frac{m}{k} d_{max})$ edges. All $O(m/k)$ passes take no more than $O(\frac{m}{k}(\frac{m}{k} d_{max})) \in O((\frac{m}{k})^2 d_{max})$ time, and if $k \in O(\sqrt{m})$ then the time is $O(md_{max})$. Thus the total time is,

$$O(m + md_{max}) \in O(md_{max})$$

If $d_{max} \in O(1)$ then the time complexity is $O(m)$. □

Listing 2: k-core algorithm

```
L ← sorted edges
count ← 1
while count ≠ 0 do
    count ← 0; degree ← 0
    create new L'
    for all (v, u) ∈ L do
        if done with edges for a v then
            if degree ≥ k then
                add T_v to L'
            else
                count ← count + 1
            end if
            create new T_v
            degree ← 0
        else
            add (u, v) to T_v
            degree ← degree + 1
        end if
    end for
    L = L'                ▷ L remains locally sorted
end while
```

MapReduce k-Core algorithm

Round 1 given $(v, (u, d_u))$ input

Map: Identity

$$\langle v, (u, d_u) \rangle \longrightarrow \langle v, (u, d_u) \rangle$$

Reduce: Output reverse edges

$$\langle v, \{(u, d_u) \mid u \in N(v)\} \rangle \longrightarrow \{\langle u, v \rangle\} \iff d_v, d_u \geq k$$

Rounds $[2..O(n)]$

Map: Identity

$$\langle v, u \rangle \longrightarrow \langle v, u \rangle$$

Reduce: Output reverse edges

$$\langle v, \{u \mid u \in N(v)\} \rangle \longrightarrow \{\langle u, v \rangle\} \iff d_v \geq k$$

Evaluation of degree

Evaluating $d_v \geq k$ is possible in $O(1)$ memory by using a k-element buffer

- reducer reads u values into buffer
- if buffer is filled then d_v is at least k so...
- output the u already in the buffer and then output all remaining u

Preliminary exercise

Experiment

Test the easy MapReduce algorithms on real datasets

Datasets

Stanford Network Analysis Project (SNAP)

as-Skitter	Autonomous Systems Internet topology graph
com-Orkut	Orkut online social network
com-Friendster	Friendster online social network

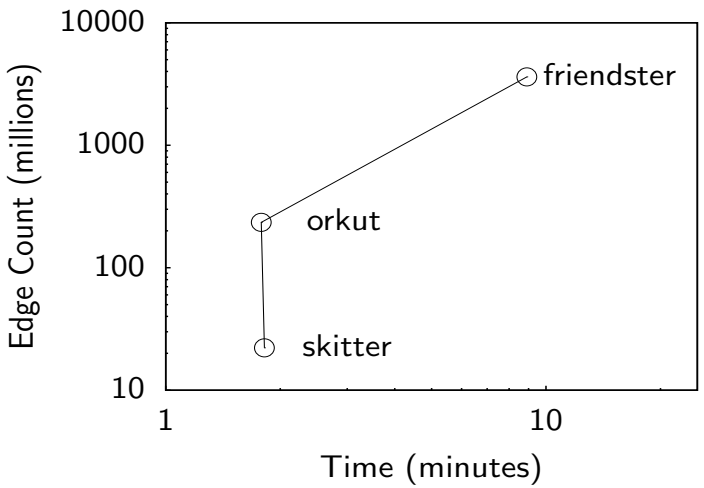
Hadoop Yarn Cluster

1200 nodes
32 cores per node
128 GB RAM per node

Connected Component Benchmarks on SNAP datasets

SNAP Graphs (undirected)

	n (vertices)	2m (edges)	# of concom
Skitter	1,696,415	22,190,596	756
Orkut	3,072,441	234,370,166	1
Friendster	65,608,366	3,612,134,270	1



	time (minutes)	rounds
Skitter	1.82	10
Orkut	1.78	7
Friendster	8.90	10

A post-step to organize labels is included

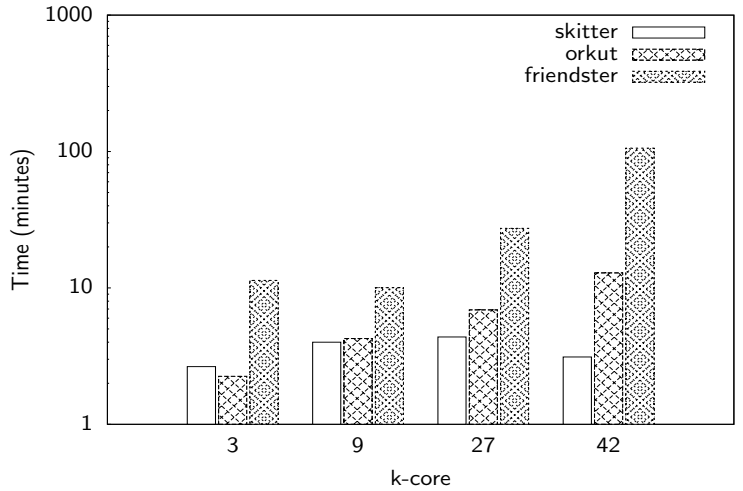
k-Core Benchmarks on SNAP datasets

SNAP Graphs (undirected)

	n (vertices)	2m (edges)	# of concom
Skitter	1,696,415	22,190,596	756
Orkut	3,072,441	234,370,166	1
Friendster	65,608,366	3,612,134,270	1

	Skitter	Orkut	Friendster
3-core	2.65	2.25	11.35
9-core	4.00	4.25	10.03
27-core	4.37	6.90	27.37
42-core	3.12	12.87	105.95

Time (minutes)



	Skitter	Orkut	Friendster
3-core	11	6	17
9-core	18	11	15
27-core	22	18	41
42-core	16	37	172

# of rounds

A pre-step round for degree annotation is included

Conclusion

Results

Demonstrated very easy algorithms under hard constraints.

- no part of the graph is stored in memory
- data access per step is single-pass and sequential
- no saved state either globally or locally — tasks perish after each step
- very minimal memory per task
- simple implementations
- room for improvement

Bibliography I

[1] A. F. Alexander-Bloch, N. Gogtay, et al.
Disrupted modularity and local connectivity of brain functional networks in childhood-onset schizophrenia.
Frontiers in Systems Neuroscience, 4(147), 2010.

[2] D. Godwin, R. L. Barry, and R. Marois.
Breakdown of the brain's functional network modularity with awareness.
PNAS, 112(12):3799–3804, March 2015.

[3] Y. He, Z. Chen, and A. Evans.
Structural insights into aberrant topological patterns of large-scale cortical networks in alzheimer's disease.
The Journal of Neuroscience, 28(18):4756–4766, 2008.

[4] M.-E. Lynall, D. S. Bassett, R. Kerwin, et al.
Funtional connectivity and brain networks in schizophrenia.
The Journal of Neuroscience, 30(29):9447–9487, 2010.

[5] R. Penfold and P. J. Wilde.
Enumerating and indexing many-body intramolecular interactions: a graph theoretic approach.
Journal of mathematical chemistry, 53(7):1634–1648, 2015.

[6] J. Scott, T. Ideker, and R. M. Karp.
Efficient algorithms for detecting signaling pathways in protein interaction networks.
Journal of Computational Biology, 13(2):133–144, 2006.

[7] S. B. Seidman.
Network structure and minimum degree.
Social Networks, 5:269–287, 1983.

[8] R. Singh, J. Xu, and B. Berger.
Pairwise global alignment of protein interaction networks by matching neighborhood topology.
In *Proceedings of the 11th annual international conference on research in computational molecular biology*, RECOMB'07, pages 16–31. Springer-Verlag, 2007.

Bibliography II

[9] S. Spivak, A. Ismagilova, and I. Khamitova.
Graph-theoretical method for determining routes of complex chemical reactions.
In *Doklady Physical Chemistry*, volume 434, pages 169–171. Springer, 2010.

[10] L. Wang, B. Stumm, and V. Helms.
Graph-theoretical identification of dissociation pathways on free energy landscapes of biomolecular interaction.
Journal of computational chemistry, 31(4):847–854, 2010.