Parallel Graph Decompositions Using Random Shifts

Gary L. Miller, Richard Peng, and Shen Chen

Presentation by Eeshan Tripathii

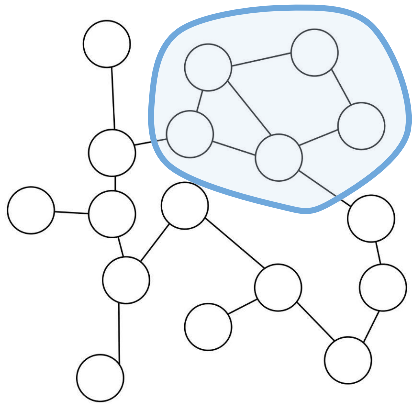
The Problem

“Decomposing an undirected unweighted graph into small diameter pieces”

Background Information

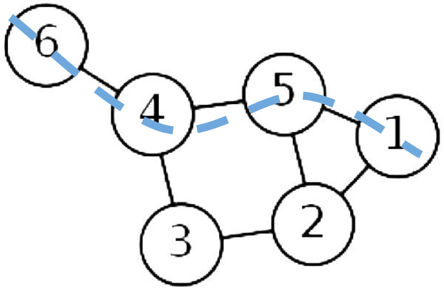
“Decomposing an undirected unweighted graph into small diameter pieces”

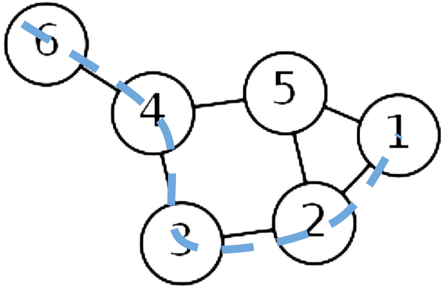
- Decomposing
  - Breaking a graph into smaller pieces such that the two sub-graphs share no edges
- Undirected
  - None of the edges in the graph have directions
- Unweighted
  - None of the edges in the graph have weights (all have weight 1)
- Diameter
  - The length of the shortest path between the farthest nodes



"Decomposing an undirected unweighted graph into small diameter pieces"

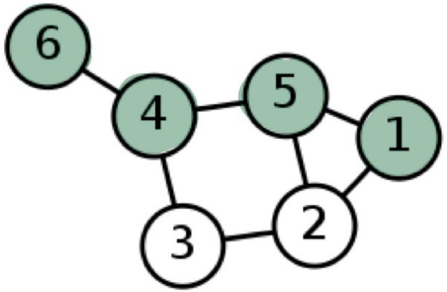
- Decomposing
  - Breaking a graph into smaller pieces such that the two sub-graphs share no edges
- Undirected
  - None of the edges in the graph have directions
- Unweighted
  - None of the edges in the graph have weights (all have weight 1)
- Diameter
  - The length of the shortest path between the farthest nodes

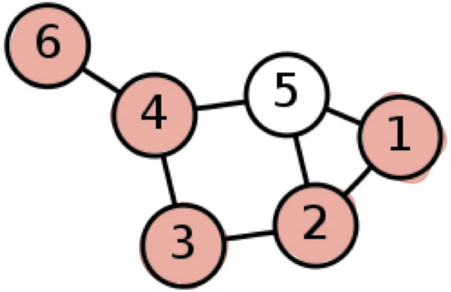




"Decomposing an undirected unweighted graph into small diameter pieces"

- Decomposing
  - Breaking a graph into smaller pieces such that the two sub-graphs share no edges
- Undirected
  - None of the edges in the graph have directions
- Unweighted
  - None of the edges in the graph have weights (all have weight 1)
- Diameter
  - The length of the shortest path between the farthest nodes





“Decomposing an undirected unweighted graph into small diameter pieces”

- Why use diameter as a parameter?
  - A variety of other measures are used
  - More intricate measures such as conductance have proven to be more useful in many applications
  - However, even algorithms that use conductance, as well as many others, use simpler low diameter decompositions as a subroutine

“Decomposing an undirected unweighted graph into small diameter pieces”

- How to compute the diameter of a graph?
  - Strong diameter
    - Restricts the shortest path between two vertices in S to only use vertices S (S being the sub-graph)
    - Parallelized with nearly-linear work
  - Weak diameter
    - Allows for shortcuts through vertices outside of S
    - Parallelized with quadratic work in the optimal tree metric embedding algorithm

“Decomposing an undirected unweighted graph into small diameter pieces”

- How to compute the diameter of a graph?
    - **Strong diameter**
        - Restricts the shortest path between two vertices in S to only use vertices S (S being the sub-graph)
        - Parallelized with nearly-linear work
    - Weak diameter
        - Allows for shortcuts through vertices outside of S
        - Parallelized with quadratic work in the optimal tree metric embedding algorithm

“Decomposing an undirected unweighted graph into small diameter pieces”

“Decomposing ***in Parallel*** *an undirected unweighted graph into small diameter pieces”*

Why?

Applications

- Generally
  - Decompositions form critical subroutines in a number of graph algorithms.
- Low Diameter Decompositions
  - Approximations to sparsest cut
  - Construction of spanners
  - Parallel approximations of shortest path in undirected graphs
  - Generating low-stretch embedding of graphs into trees
  - Construction of low-stretch spanning trees
  - Computing separators in minor-free graphs
  - Nearly linear work parallel solvers for SDD linear systems

Applications

- Generally
  - Decompositions form critical subroutines in a number of graph algorithms.
- Low Diameter Decompositions
  - Approximations to sparsest cut
  - Construction of spanners
  - Parallel approximations of shortest path in undirected graphs
  - Generating low-stretch embedding of graphs into trees
  - Construction of low-stretch spanning trees
  - Computing separators in minor-free graphs
  - **Nearly linear work parallel solvers for SDD linear systems**

SDD Linear Systems

- Low diameter graph decompositions using strong diameter as a measure are particularly useful for solving symmetric diagonally dominant linear systems
- Computing maximum flow and negative length shortest paths
- Used in many applications
    - Symmetric matrix where one where $|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \quad \text{for all } i$

SDD Linear Systems

- Low diameter graph decompositions using strong diameter as a measure are particularly useful for solving symmetric diagonally dominant linear systems
- Computing maximum flow and negative length shortest paths
- Used in many applications
  - Symmetric matrix where one where $|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \quad \text{for all } i$

$$\begin{bmatrix} 3 & 2 & 1 \\ 2 & -3 & 0 \\ 1 & 0 & 5 \end{bmatrix}$$

$$|+3| \geq |+2| + |+1|$$

$$|-3| \geq |+2| + |0|$$

$$|+5| \geq |+1| + |0|$$

SDD Linear Systems

Algorithms solving symmetric diagonally dominant linear systems created by authors of this paper

Richard Peng
M.I.T.
rpeng@mit.edu

https://dl.acm.org/doi/pdf/10.1145/2591796.2591832

Gary L. Miller
Carnegie Mellon University
glmiller@cs.cmu.edu

Richard Peng
M.I.T.
rpeng@mit.edu

https://www.cs.cmu.edu/~glmiller/Publications/Papers/CKMPPRX14.pdf

Previous Approaches

Relevant Research

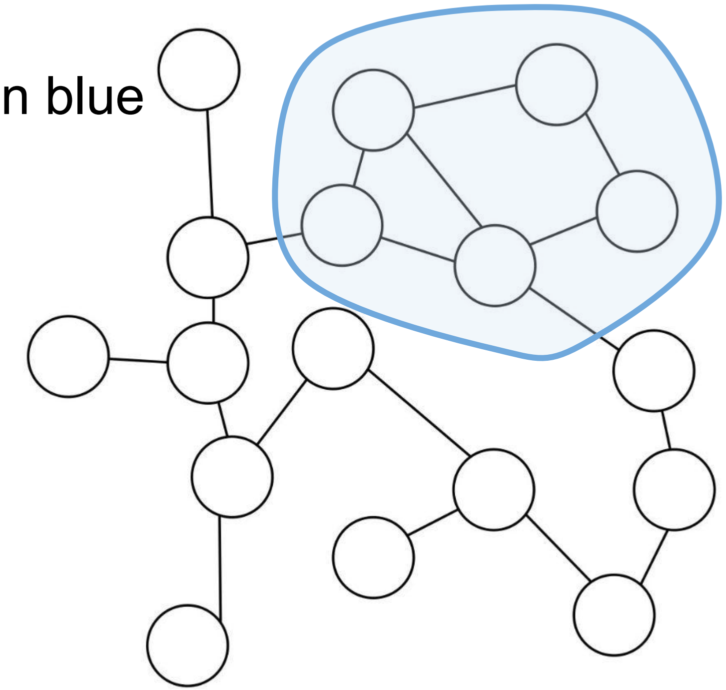
- Previous algorithms based upon conductance rather than diameters have studied
  - This algorithm could be used as a subroutine for them
- Others have used diameters but their work was either serial or measuring diameters weakly
- Shifted shortest path approach introduced in [Blelloch, Gupta, Koutis, Miller, Peng, Tangwongsan, SPAA 2011]
  - This algorithm is largely based on this work and mainly seeks to simply it while maintaining the same asymptotic runtimes

Overview of Algorithm

Ball Growing

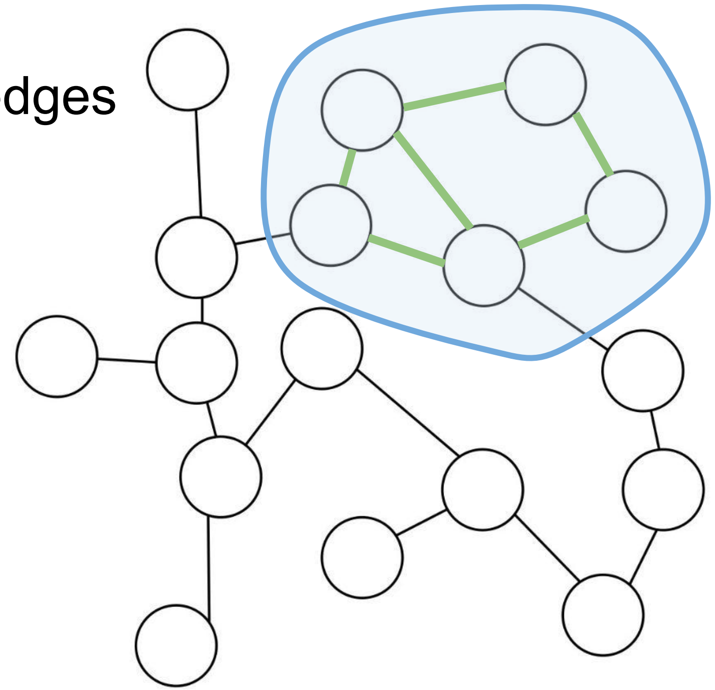
Internal Edges vs External Edges

Consider the subgraph in blue



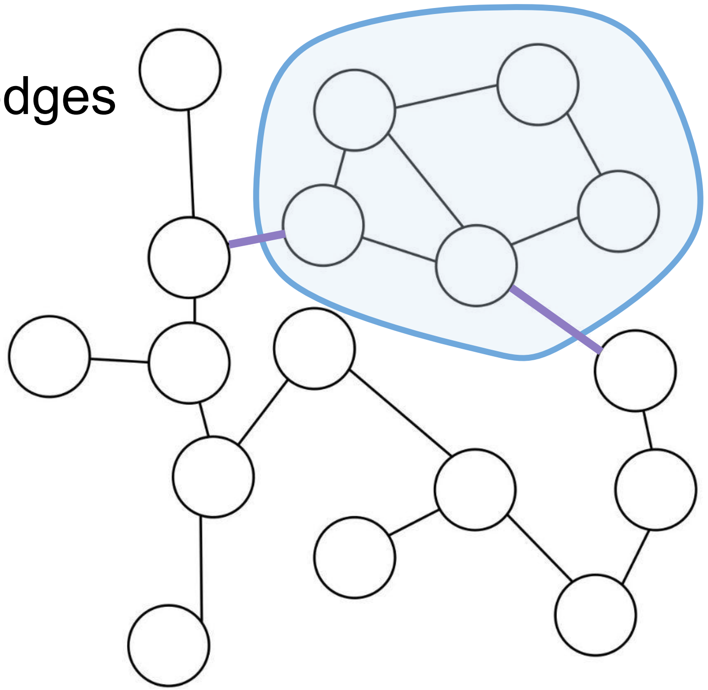
Internal Nodes vs External Nodes

These are the internal edges



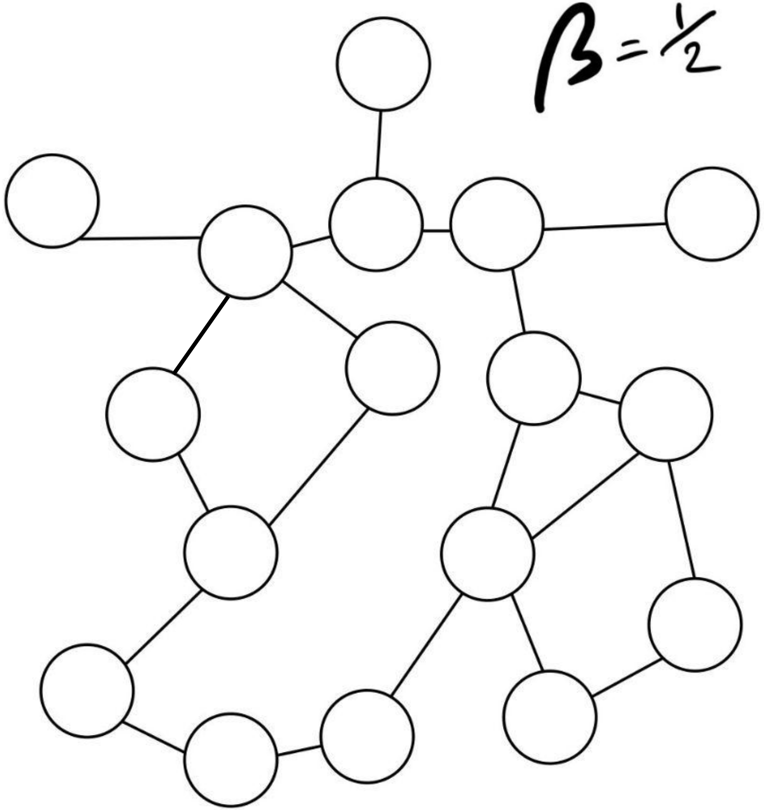
Internal Nodes vs External Nodes

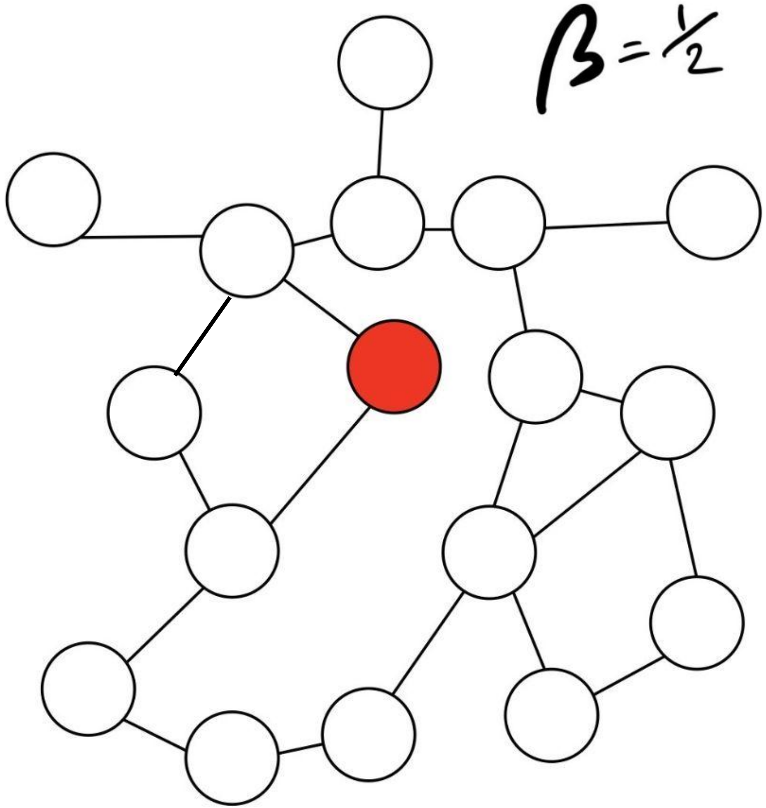
These are the external edges



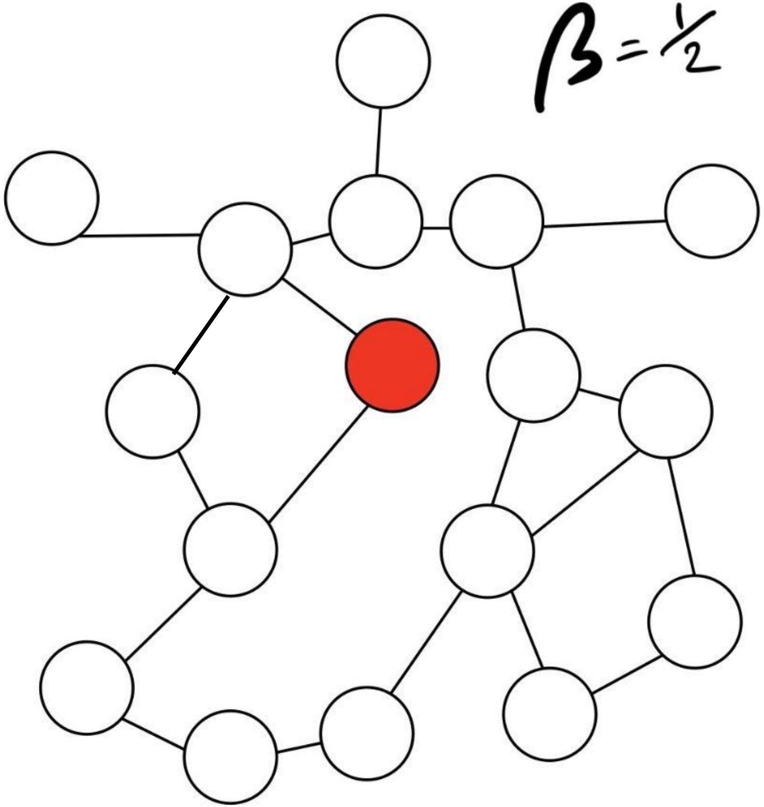
$$\text{Constriction is defined as} = \frac{\text{the number of external edges}}{\text{the number of internal edges}}$$

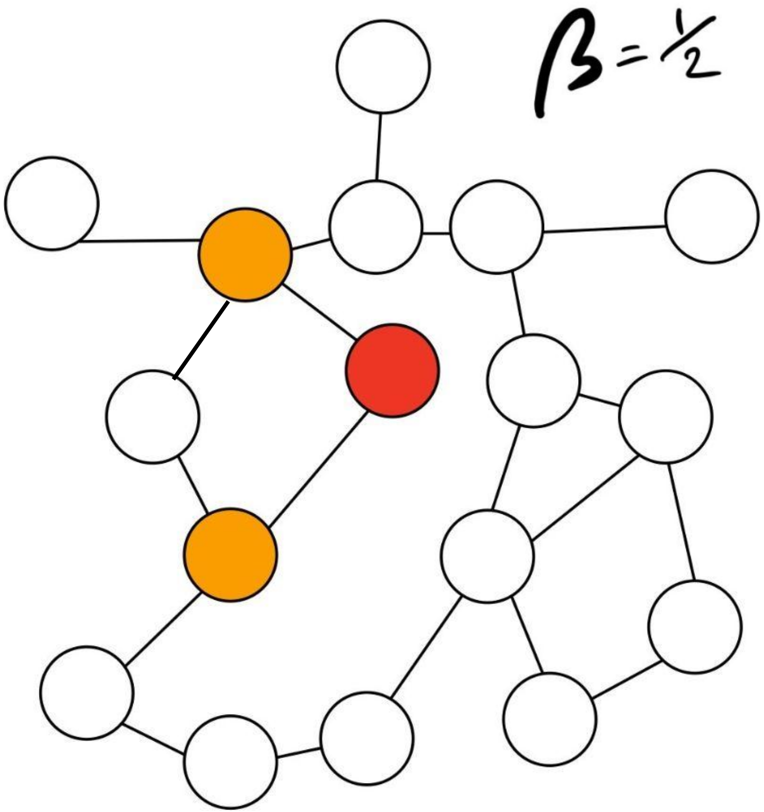
Starts with a single vertex, and repeatedly adds the neighbors similarly to BFS.
It terminates when the constriction is less than β.


β = 1/2

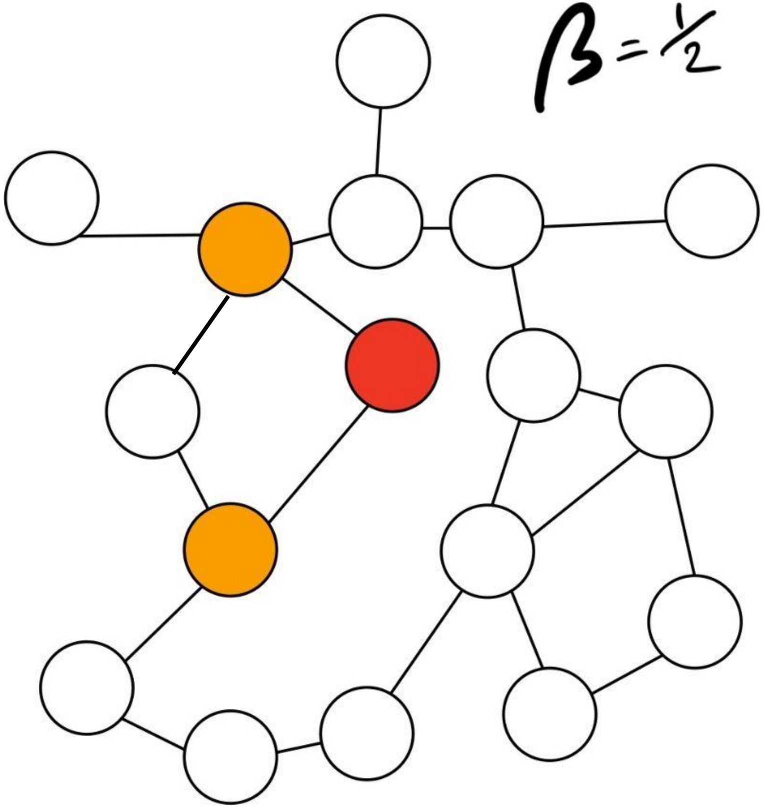

β = 1/2

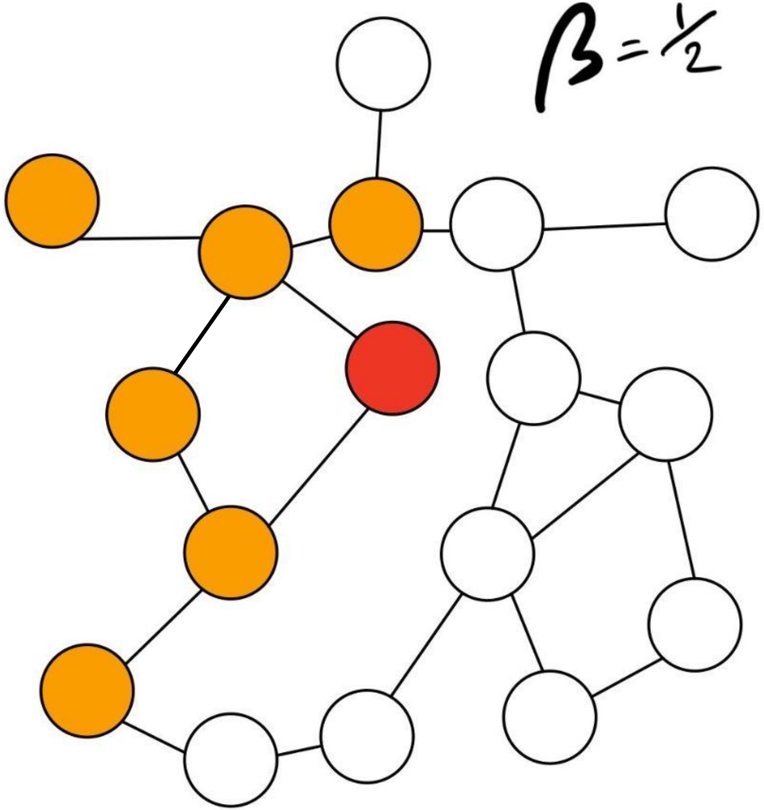
External edges: 2
Internal edges: 0
Constriction: 2/0



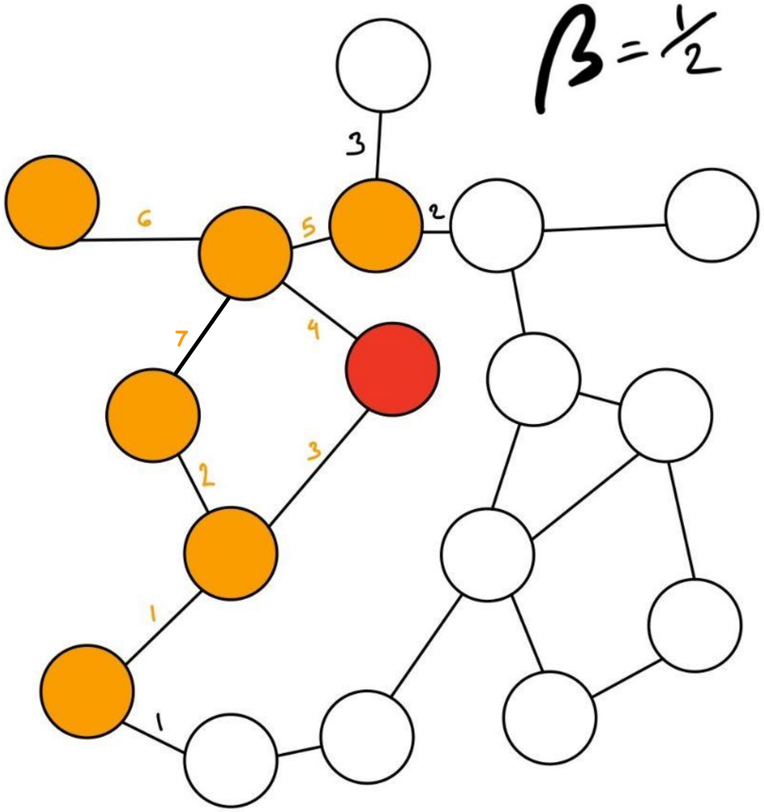

$\beta = 1/2$

External edges: 5
Internal edges: 2
Constriction: 5/2



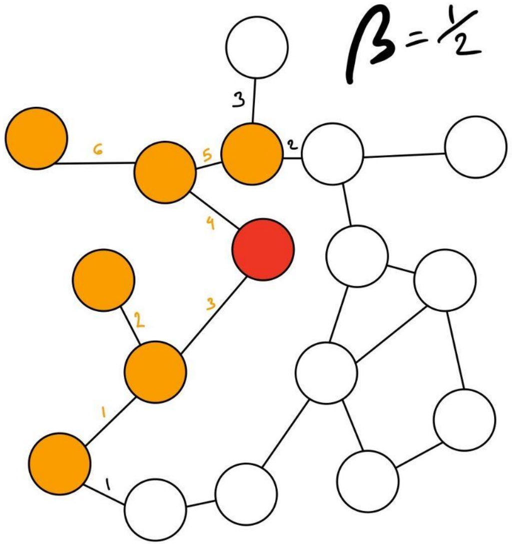

β = 1/2

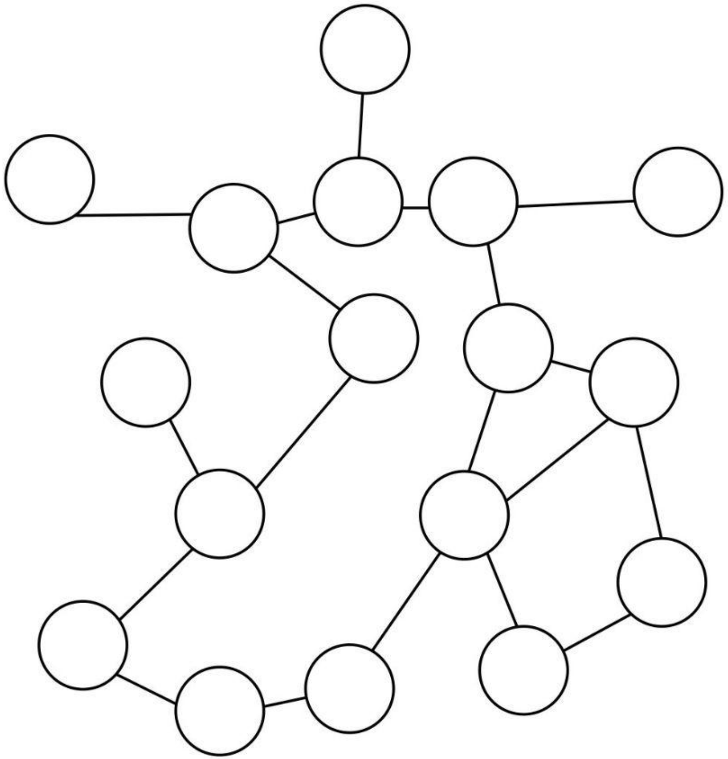
External edges: 3
Internal edges: 7
Constriction: 3/7 < 1/2

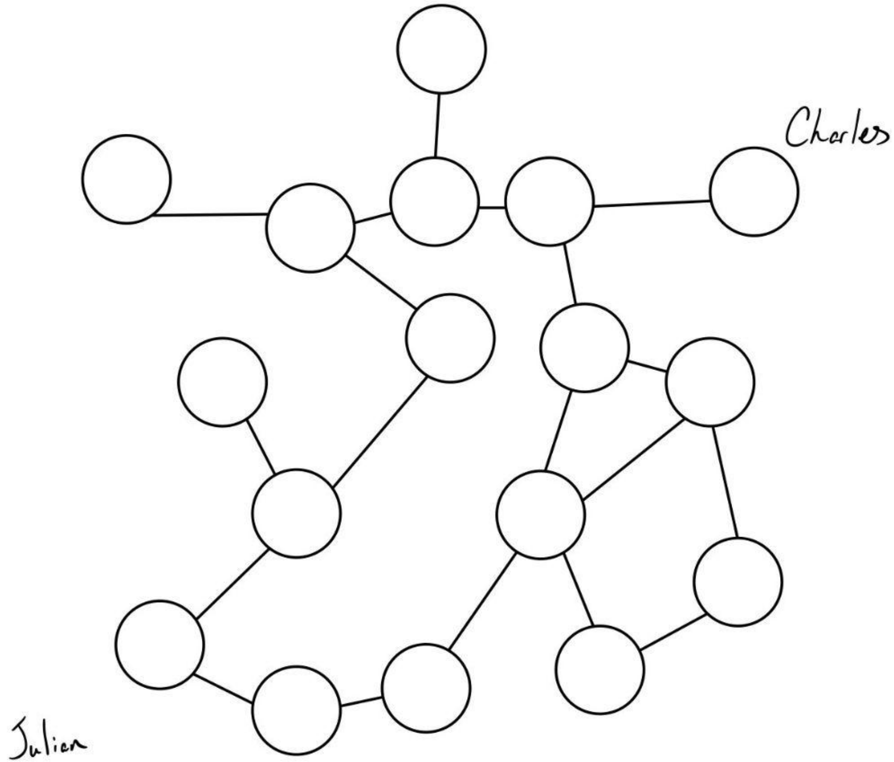


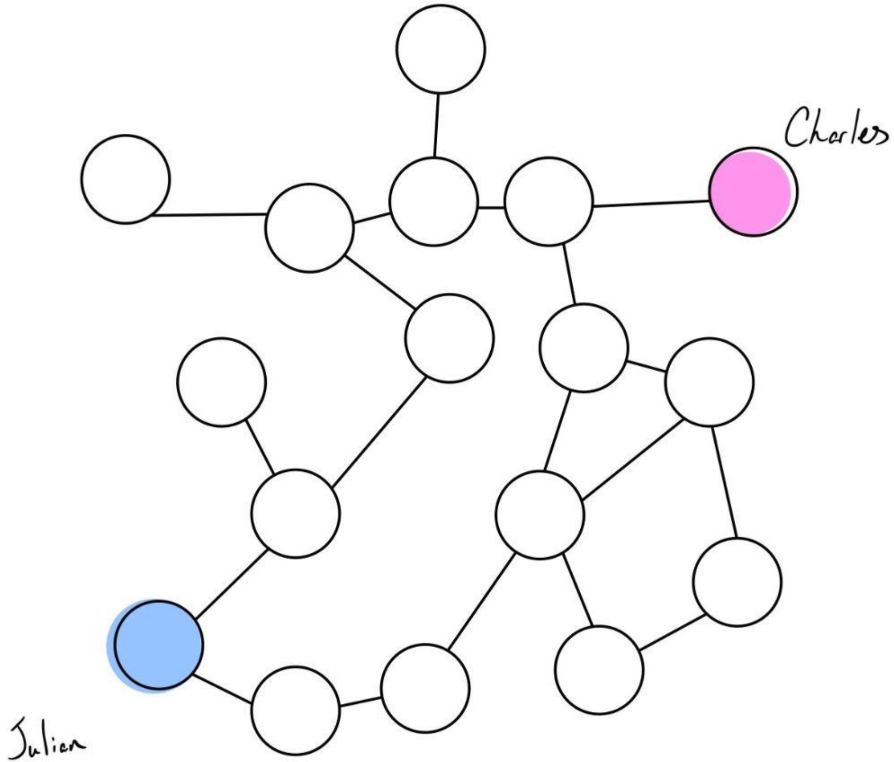
Ball Growing

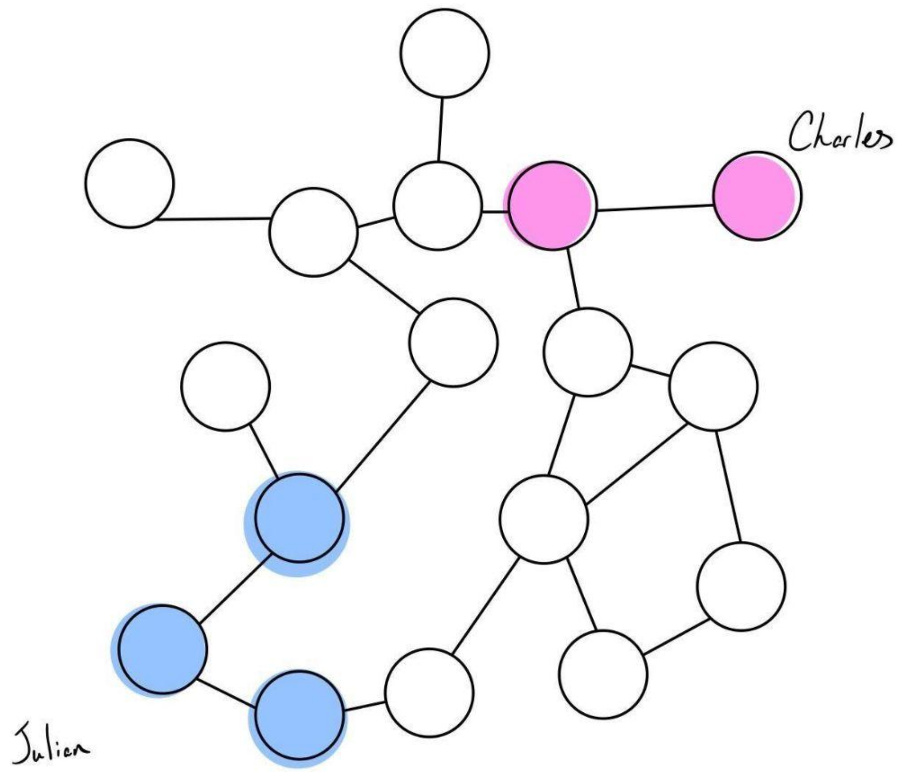
- Diameter of a piece is bounded by $O(\frac{\log n}{\beta})$
- Easy to run serially
  - Find the second subgraph after we are done finding the first
- However, if we parallelize then we get problems with overlapping

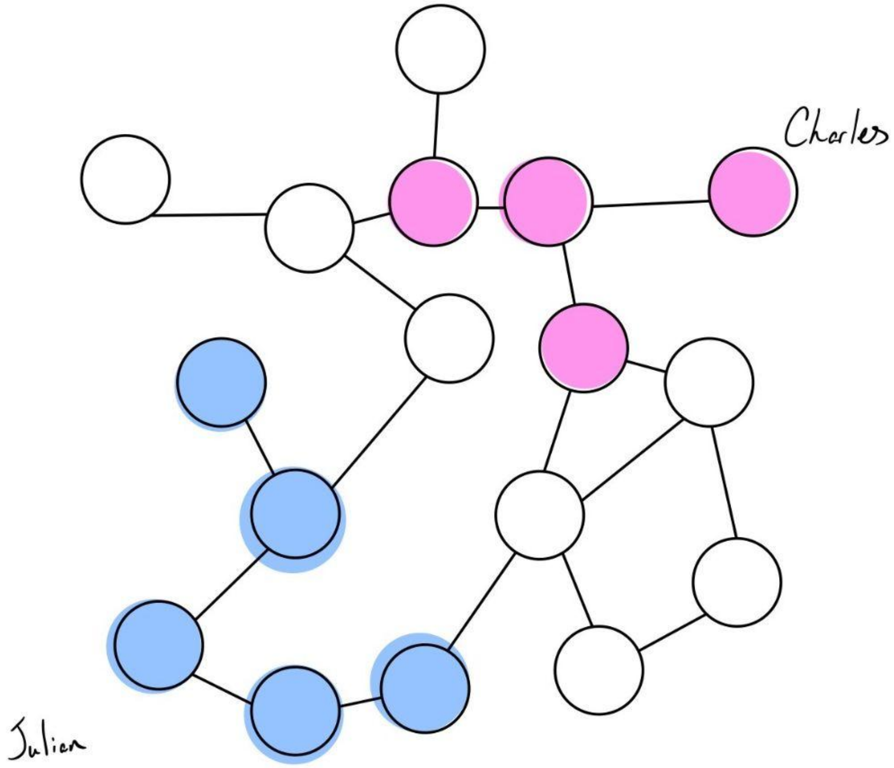


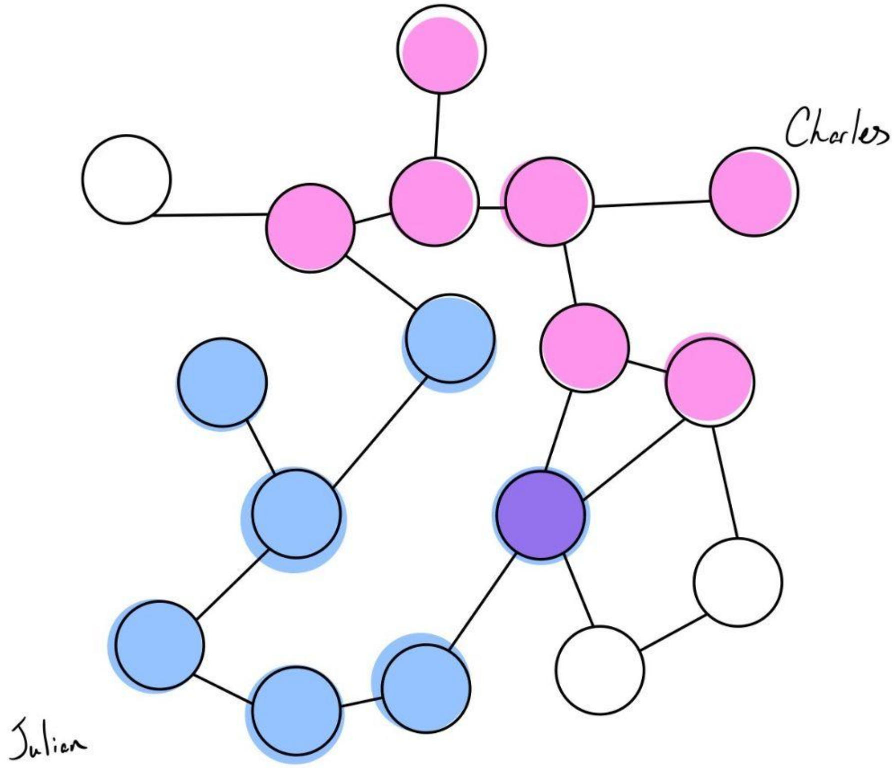



Charles
Julien

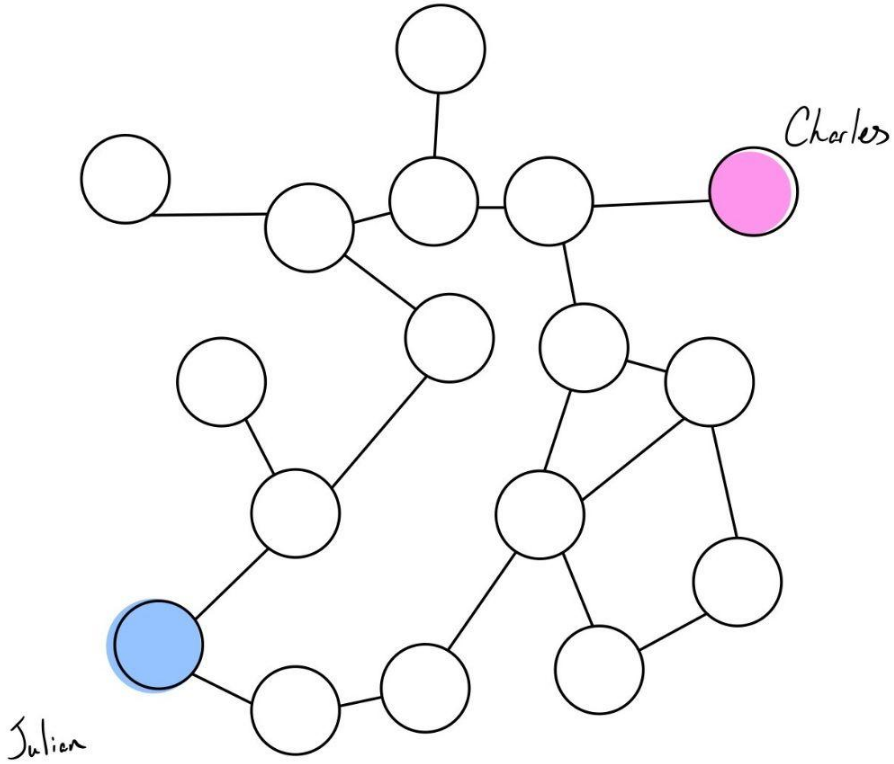

Charles
Julien


Charles
Julien


Charles
Julien



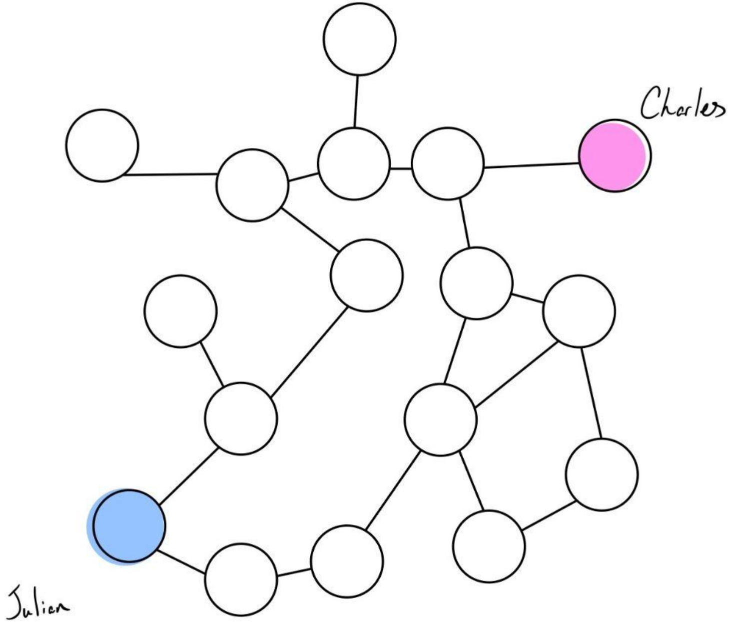
Charles
Julien


Charles
Julien

Shifting

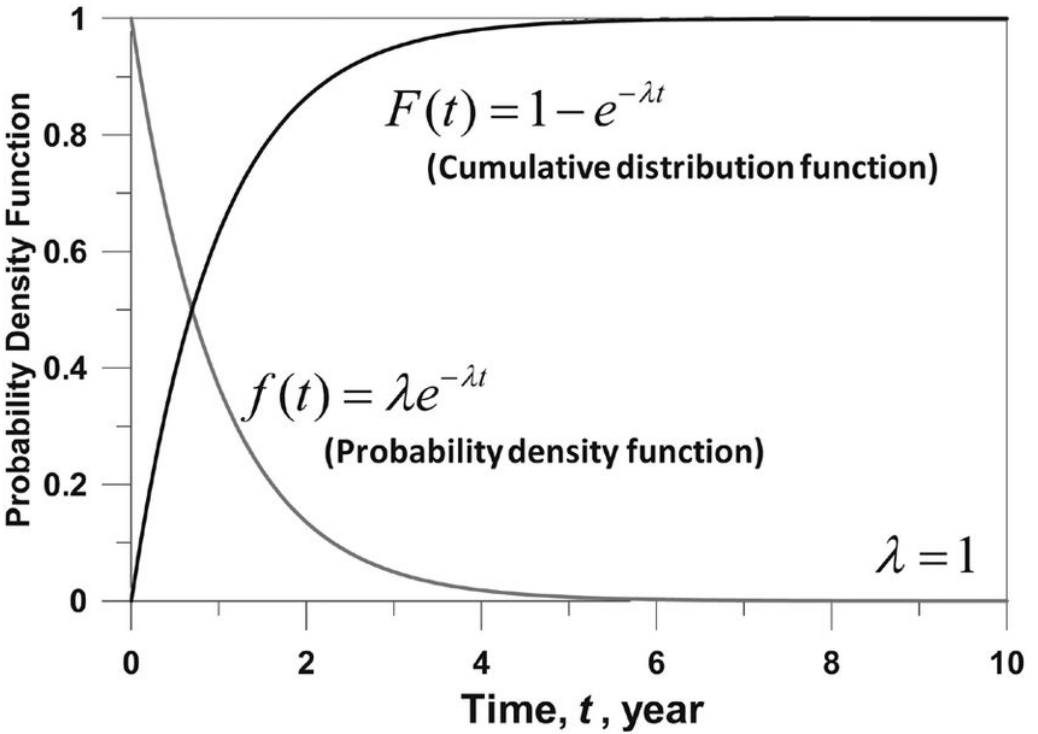
Dealing with Overlaps

```
Decompose(V):
  cilk_for(u in V):
    ball_growing(u, rand_time(node))

ball_growing(u, start_time):
  if time == start_time:
    if !u.cluster:
      u.cluster = u
      BFS(u)

BFS(u):
  cilk_for(v in u.neighbors):
    if !v.cluster:
      v.cluster = u.cluster
      BFS(v)
```



Distances not Times

$$\mathrm{dist}_{-\delta}(u, v) = \mathrm{dist}(u, v) - \delta_u$$



$$F_{Exp}(x, \gamma) = \mathbf{Pr}\left[Exp(\gamma) \leq x\right] = \begin{cases} 1 - \exp(-\gamma x) & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

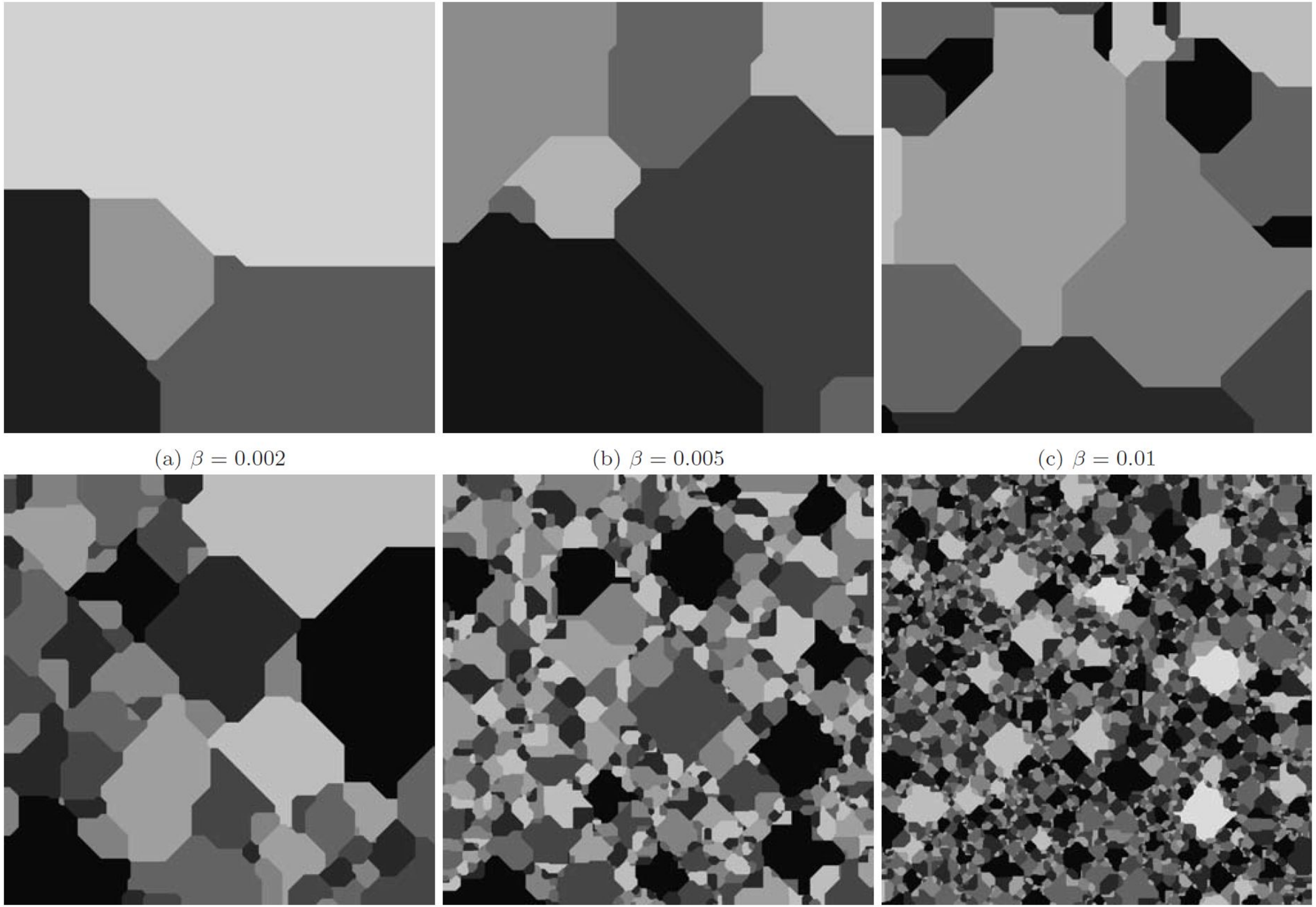
Algorithm 1 Parallel Partition Algorithm

PARALLEL PARTITION

Input: Undirected, unweighted graph $G = (V, E)$, parameter $0 < \beta < 1$

Output: $(\beta, O(\log n/\beta))$ decomposition of G WHP

1: *IN PARALLEL* each vertex u picks δ_u independently from an exponential distribution with mean $1/\beta$.
2: *IN PARALLEL* compute $\delta_{\max} = \max\{\delta_u \mid u \in V\}$
3: Perform *PARALLEL BFS*, with vertex u starting when the vertex at the head of the queue has distance more than $\delta_{\max} - \delta_u$.
4: *IN PARALLEL* Assign each vertex u to point of origin of the shortest path that reached it in the BFS.



(a) $\beta = 0.002$ (b) $\beta = 0.005$ (c) $\beta = 0.01$

(d) $\beta = 0.02$ (e) $\beta = 0.05$ (f) $\beta = 0.1$

1000x1000 grid

Impact and Analysis

- By picking shifts uniformly from a sufficiently large range, a $(\beta, O(\frac{\log^c n}{\beta}))$ decomposition can be obtained.
- A common algorithmic routine is to partition a graph into O(log n) blocks such that each connected piece in a block has diameter O(log n)
  - This can be obtained using this algorithm by running a (1/2, O(log n)) low diameter decomposition O(log n) times as the number of edges not in a block decreases by a factor of 2 per iteration
- As a sequential algorithm, it can also lead to similar guarantees on weighted graphs to Bartal's decomposition scheme as well as generalizations needed for improved low stretch spanning tree algorithms
- Parallel performance with weighted graph has not been analyzed

Future Steps

- Obtaining similar parallel guarantees in the weighted setting
- Showing clustering-based properties