Parallel Graph Decompositions Using Random Shifts

Gary L. Miller, Richard Peng, and Shen Chen Xu
CMU

Published at SPAA’13

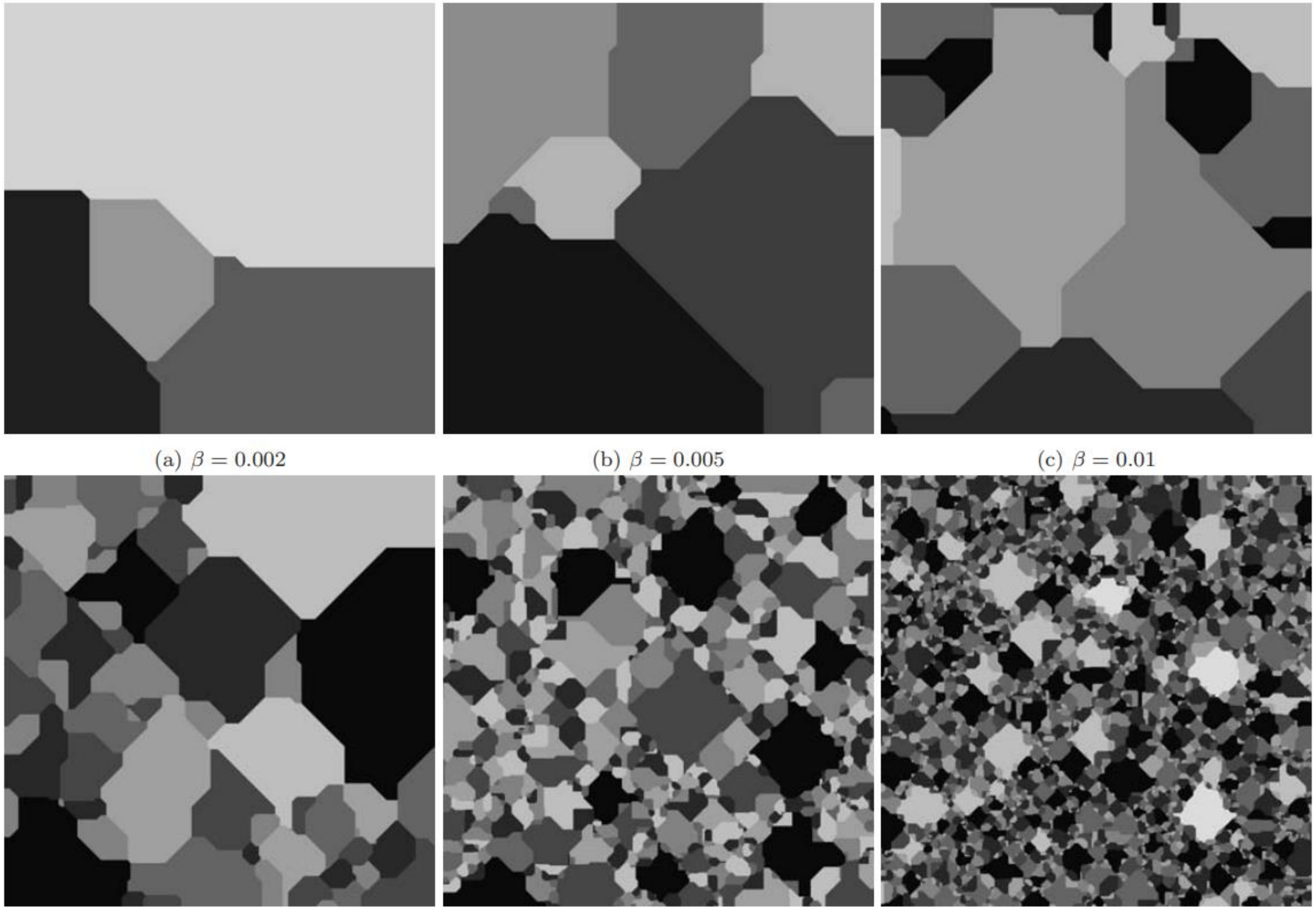
Presented by Victor Ying

6.886 – Februrary 25, 2021

The problem

- Given unweighted, undirected, connected graph and a parameter $0 < \beta < 1$, partition the vertices such that:
  - No more than some small fraction β of edges are cut
  - Each partition induces a connected subgraph with diameter $O(\beta^{-1} \log n)$
- One typically choses β to be fairly small, e.g., $\sim 1/(\log n)^c$
- Applications:
  - Build a low-stretch spanning tree (by computing spanning tree within each partition and then joining them up)
  - Solve symmetric diagonally dominant (SDD) systems of linear equations with ε-accuracy

Examples



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

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

Figure 1: Decompositions generated by our algorithm on a 1000×1000 grid under varying values of β. Different colors represent different clusters

Sequential algorithm

- Repeat until no vertices are left:
  - Pick an arbitrary vertex v
  - Use BFS to grow a ball centered at v,
    until (# edges on boundary) < $\beta \cdot$ (# edges inside)
  - All vertices inside the ball are assigned to a new partition
    and are deleted from the graph

Improvements in complexity bounds

runs in O(m) time Prior parallel algorithm [SPAA’11] runs in $O(m \log^2 n)$ expected work and $O(\beta^{-1} \log^2 n)$ expected depth. This	Work	Span
Sequential algorithm	O(m)	O(m)
Prior parallel algorithm [SPAA’11]	$O(m \log^2 n)$ expected	$O(\beta^{-1} \log^2 n)$ expected
This work [SPAA’13]	O(m) expected	$O(\beta^{-1} \log^2 n)$ expected

The new parallel algorithm

Algorithm 1 Parallel Partition Algorithm

PARALLEL PARTITION

Input: Undirected, unweighted graph $G = (V, E)$, parameter $0 < \beta < 1$ and parameter d indicating failure probability.

Output: $(\beta, O(\log n/\beta))$ decomposition of G with probability at least $1 - n^{-d}$.

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.

Correctness criteria

- Each partition induces a subgraph with diameter O(β-1 log n)
- No more than some small fraction β of edges can be cut
- Note: Each criterion can be cheaply verified, so if the probabilistic algorithm fails, it can be re-run. So, if a single run succeeds with high probability, that is sufficient.

The new parallel algorithm (restated)

Algorithm 2 Partition Algorithm Using Exponentially Shifted Shortest Paths

PARTITION

Input: Undirected, unweighted graph $G = (V, E)$, parameter β and parameter d indicating failure probability.

Output: $(\beta, O(\log n/\beta))$ decomposition of G with probability at least $1 - n^{-d}$.

1: For each vertex u, pick δ_u independently from $Exp(\beta)$

2: Compute S_u by assigning each vertex v to the vertex that minimizes $\text{dist}_{-\delta}(u, v)$, breaking ties lexicographically

3: **return** $\{S_u\}$

where

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

Partition diameter $O(\beta^{-1} \log n)$

- Diameter of any partition is at most $2 \cdot \delta_u$, where u is the center vertex.
- Paper lemma 4.2 says, with high probability,

$$\delta_u < (d + 1)\, \beta^{-1} \log n$$

  for all vertices u.
- Proof sketch: simply compute the CDF of the exponential distribution to see each vertex has vanishingly tiny probability of picking larger δ_u, then apply the union bound (Boole's inequality).
- Basically, tail of an exponential distribution cuts off pretty fast.

Each edge has probability of cut < β

- Proof sketch:
  - Consider an arbitrary edge *uv*.
  - Imagine that edge is replaced with two edges *uw* and *wv* of weight 0.5, where *w* is a new vertex at the midpoint.
  - If *u* is in partition with center u' and *v* is in partition with center v', then dist(u', w) and dist(v', w) must differ by less than 1.
  - Probability of this happening can be bounded: consider picking n independent samples from an exponential distribution, and adding a predetermined offset to each sample. What is the chance that the largest two resulting values picked fall close together?
  - Turns out this probability is < β (Lemma 4.4 & Corollary 4.5)
  - If each individual edge has probability < β of being cut, then with high probability the total fraction of edges cut is < β

Work and span

Algorithm 1 Parallel Partition Algorithm

PARALLEL PARTITION

Input: Undirected, unweighted graph $G = (V, E)$, parameter $0 < \beta < 1$ and parameter d indicating failure probability.

Output: $(\beta, O(\log n/\beta))$ decomposition of G with probability at least $1 - n^{-d}$.

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.

Work	Span
O(n)	O(1)
O(n)	O(log n)
O(m)	O(Δ log n) Since $\Delta = O(\beta^{-1} \log n)$, this is $O(\beta^{-1} \log^2 n)$

Practical implementation?

Algorithm 1 Parallel Partition Algorithm

PARALLEL PARTITION

Input: Undirected, unweighted graph $G = (V, E)$, parameter $0 < \beta < 1$ and parameter d indicating failure probability.
Output: $(\beta, O(\log n/\beta))$ decomposition of G with probability at least $1 - n^{-d}$.

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.

Generating real values from an exponential distribution is doable but isn't cheap

Further thoughts

- Empirical evaluation of actual implementation?
- What about weighted graphs?
  - Any analysis would need a bound in the variation among edge weights
- What about other decomposition quality criteria?
  - This paper wanted partitions with low “strong diameter” (i.e., diameter of induced subgraph), but other applications only need low “weak diameter” (i.e., longest shortest path between vertices in a partition, where the path is allowed to take shortcut through other partitions)