Contact Info:

- email: yangpliu@stanford.edu
- website: *yangpliu.github.io*

arXiv: 1907.10779

Constant Girth Approximation for Directed Graphs in Subquadratic Time

Shiri Chechik, Yang P. Liu, Omer Rotem, Aaron Sidford




Talk Outline

Part I:
Background

Talk Outline

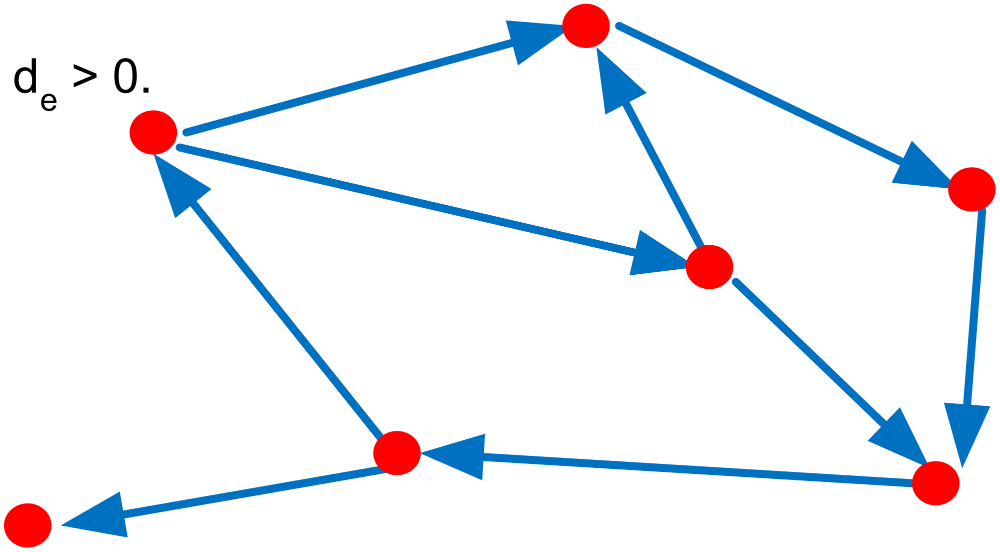
Part I: Background → Part II: Algorithms

Approximation Algorithms for the Girth

Approximation Algorithms for the Girth

Directed Graph G = (V,E)

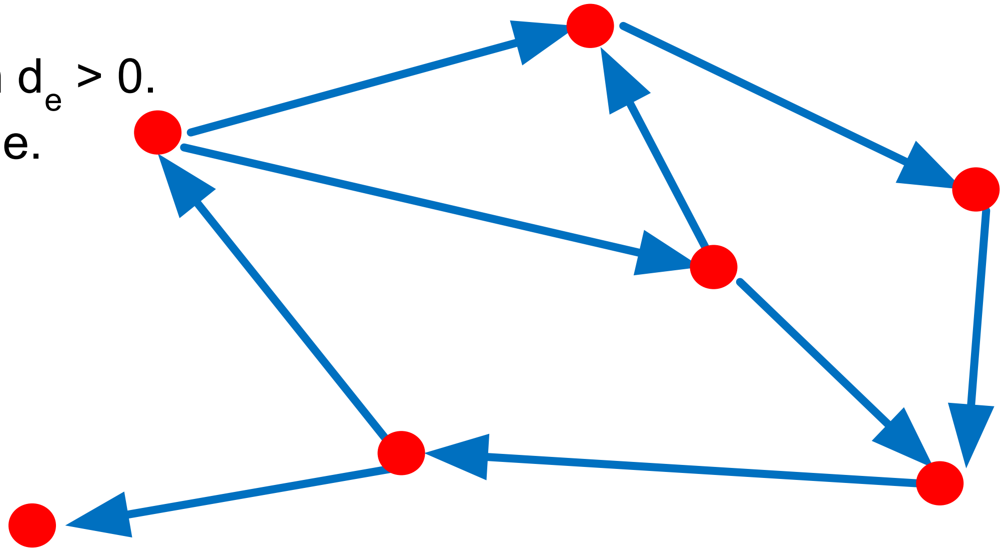
- n vertices, m edges
- Edge e = (u -> v) has length $d_e > 0$.



Approximation Algorithms for the Girth

Directed Graph G = (V,E)

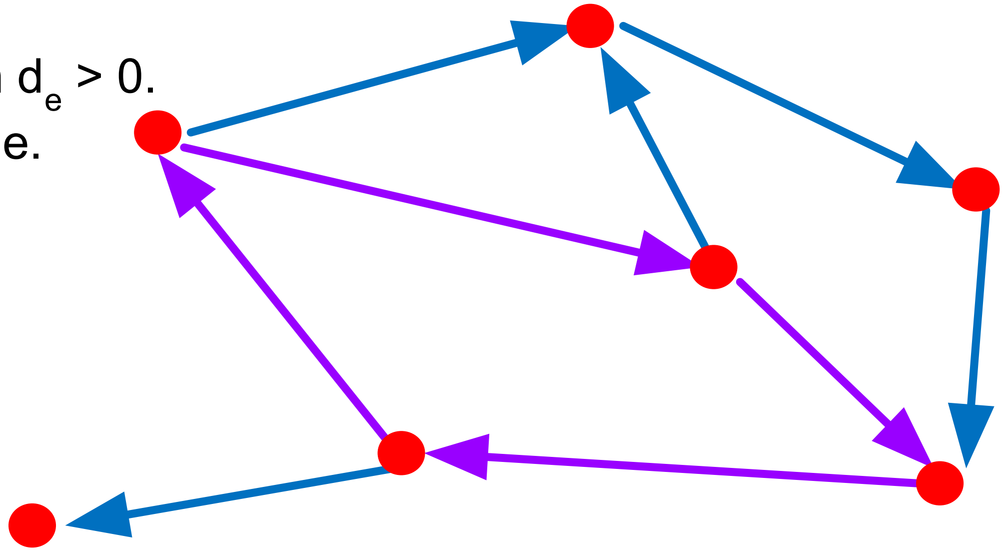
- n vertices, m edges
- Edge e = (u -> v) has length $d_e > 0$.
- Girth = minimum length cycle.



Approximation Algorithms for the Girth

Directed Graph G = (V,E)

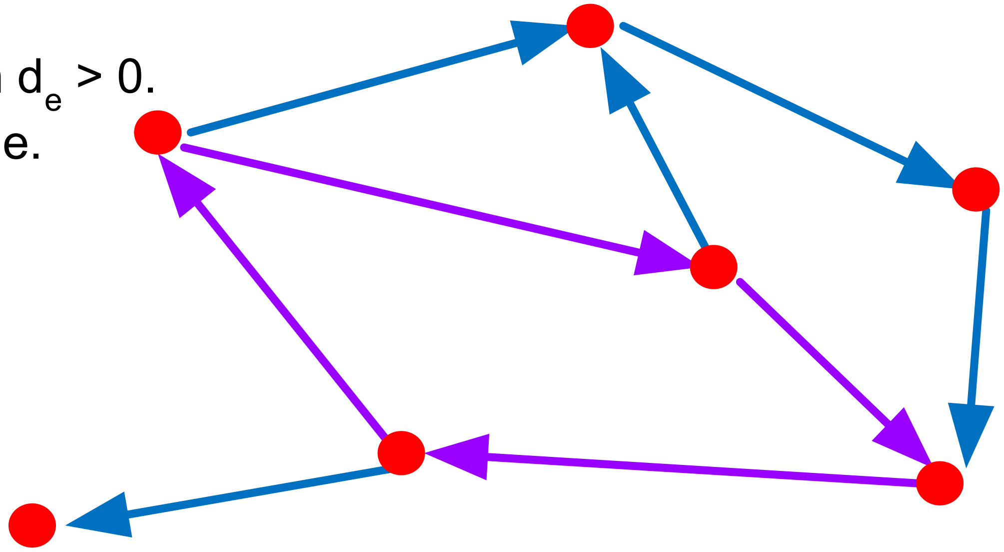
- n vertices, m edges
- Edge e = (u -> v) has length $d_e > 0$.
- Girth = minimum length cycle.



Approximation Algorithms for the Girth

Directed Graph G = (V,E)

- n vertices, m edges
- Edge e = (u -> v) has length $d_e > 0$.
- Girth = minimum length cycle.



Approximation Algorithms

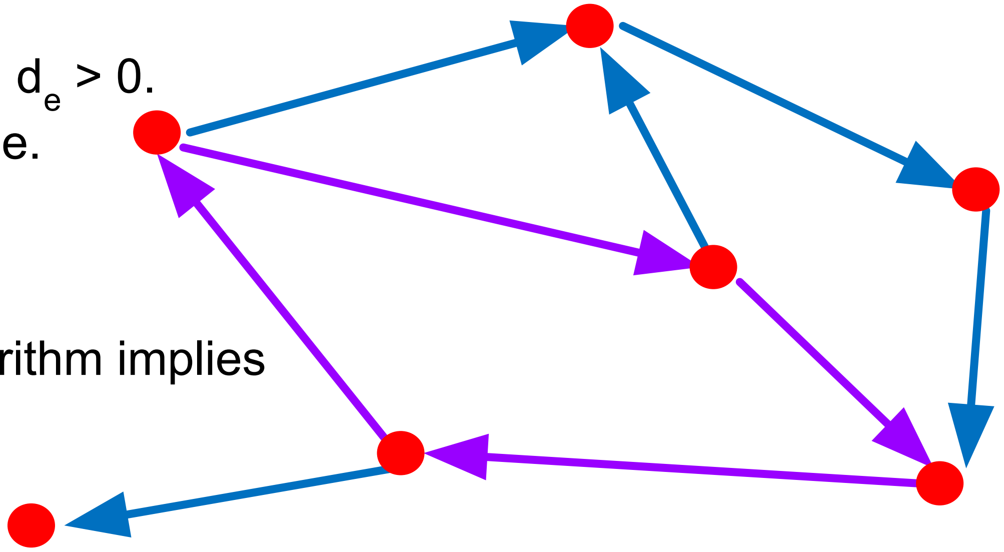
Approximation Algorithms for the Girth

Directed Graph G = (V,E)

- n vertices, m edges
- Edge e = (u -> v) has length $d_e > 0$.
- Girth = minimum length cycle.

Approximation Algorithms

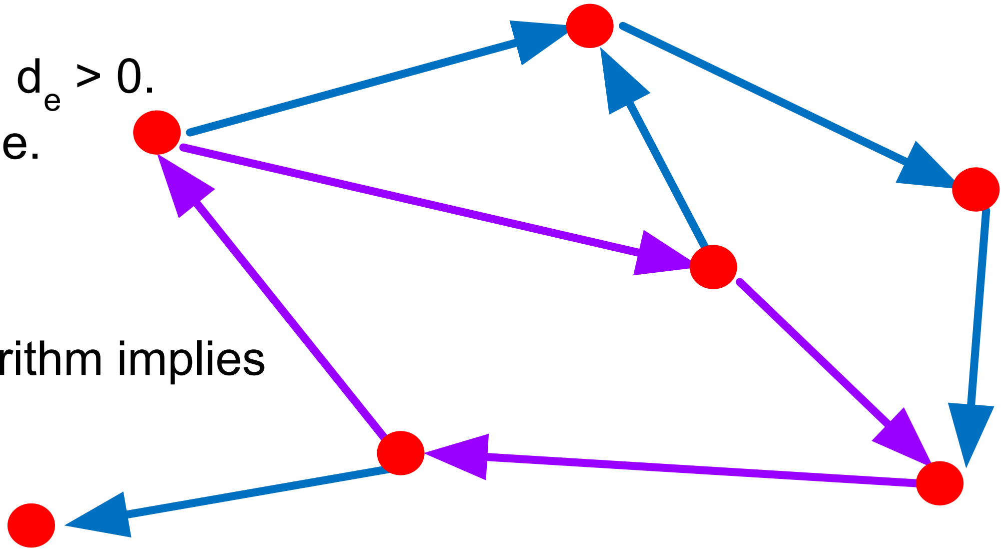
- [WW10] $n^{3-\varepsilon}$ time exact algorithm implies subcubic APSP.



Approximation Algorithms for the Girth

Directed Graph G = (V,E)

- n vertices, m edges
- Edge e = (u -> v) has length $d_e > 0$.
- Girth = minimum length cycle.



Approximation Algorithms

- [WW10] $n^{3-\varepsilon}$ time exact algorithm implies subcubic APSP.
- Focus on multiplicative approximation algorithms.

Distance Approximation and Spanners

Distance Approximation and Spanners

Undirected Graphs

Known: (2k-1) multiplicative girth approximation in time $O(mn^{1/k})$ for integers k.

Distance Approximation and Spanners

Undirected Graphs

Known: (2k-1) multiplicative girth approximation in time $O(mn^{1/k})$ for integers k.

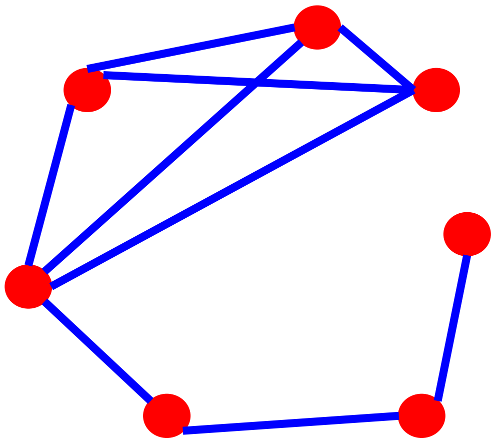
c-Spanner for G: subgraph H of G with $dist_G(u,v) <= dist_H(u,v) <= c*dist_G(u,v)$.

Distance Approximation and Spanners

Undirected Graphs

Known: (2k-1) multiplicative girth approximation in time $O(mn^{1/k})$ for integers k.

c-Spanner for G: subgraph H of G with $dist_G(u,v) <= dist_H(u,v) <= c*dist_G(u,v)$.

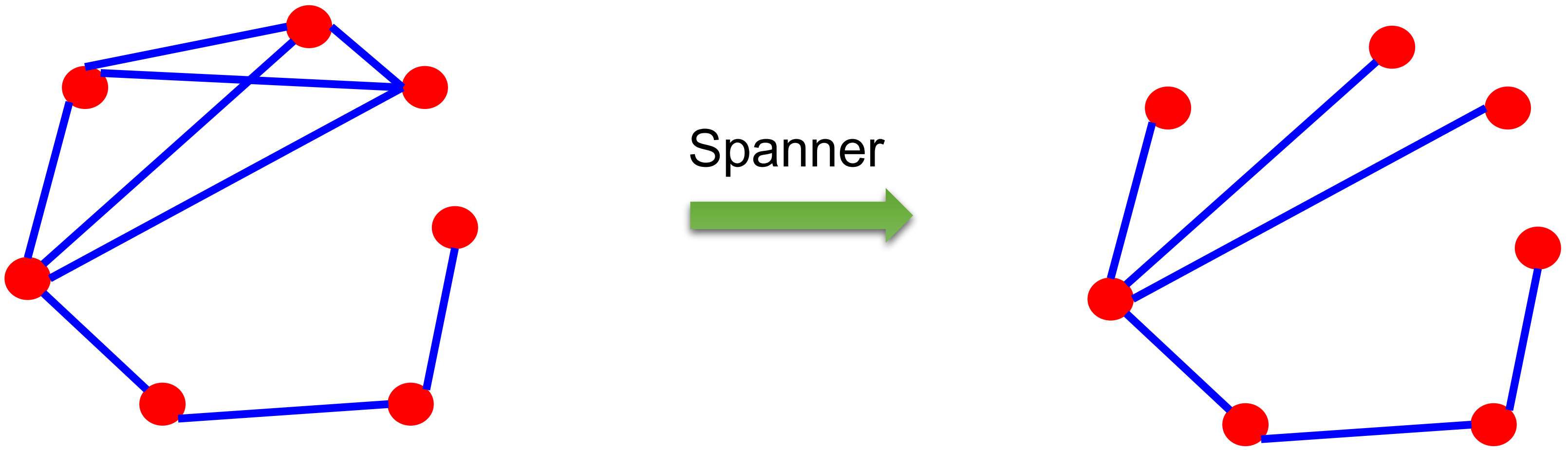


Distance Approximation and Spanners

Undirected Graphs

Known: (2k-1) multiplicative girth approximation in time $O(mn^{1/k})$ for integers k.

c-Spanner for G: subgraph H of G with $dist_G(u,v) <= dist_H(u,v) <= c*dist_G(u,v)$.



Distance Approximation and Spanners

Undirected Graphs

Known: (2k-1) multiplicative girth approximation in time $O(mn^{1/k})$ for integers k.

c-Spanner for G: subgraph H of G with $dist_G(u,v) <= dist_H(u,v) <= c*dist_G(u,v)$.

Distance Approximation and Spanners

Undirected Graphs

Known: (2k-1) multiplicative girth approximation in time $O(mn^{1/k})$ for integers k.

c-Spanner for G: subgraph H of G with $dist_G(u,v) <= dist_H(u,v) <= c*dist_G(u,v)$.

Known: Every graph has a (2k-1)-spanner with $O(n^{1+1/k})$ edges, $O(mn^{1/k})$ time.

Conjectured to be tight.

Distance Approximation and Spanners

Undirected Graphs

Known: (2k-1) multiplicative girth approximation in time $O(mn^{1/k})$ for integers k.

c-Spanner for G: subgraph H of G with $dist_G(u,v) <= dist_H(u,v) <= c*dist_G(u,v)$.

Known: Every graph has a (2k-1)-spanner with $O(n^{1+1/k})$ edges, $O(mn^{1/k})$ time.

Conjectured to be tight.

Very Similar!

Distance Approximation and Spanners

Undirected Graphs

Known: (2k-1) multiplicative girth approximation in time $O(mn^{1/k})$ for integers k.

c-Spanner for G: subgraph H of G with $dist_G(u,v) <= dist_H(u,v) <= c*dist_G(u,v)$.

Known: Every graph has a (2k-1)-spanner with $O(n^{1+1/k})$ edges, $O(mn^{1/k})$ time.

Conjectured to be tight.

[LL09,RW12,DKS17] -- best girth approximation algorithms
do not use spanners directly.

Very Similar!

Roundtrip Distance and Spanners

Roundtrip Distance and Spanners

Directed Graphs

Roundtrip Distance and Spanners

Directed Graphs

Question: do directed graphs have spanners?

Roundtrip Distance and Spanners

Directed Graphs

Question: do directed graphs have spanners?

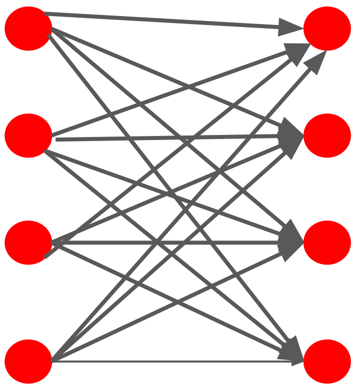
Answer: No, not in the sense described earlier!

Roundtrip Distance and Spanners

Directed Graphs

Question: do directed graphs have spanners?

Answer: No, not in the sense described earlier!



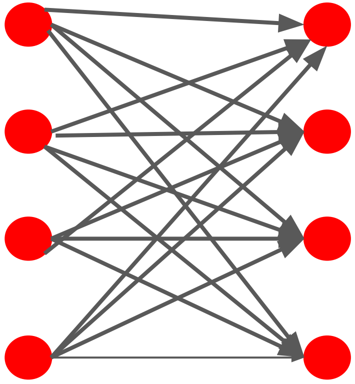
Complete bipartite directed graph has $n^2/4$ edges, but no spanners!

Roundtrip Distance and Spanners

Directed Graphs

Question: do directed graphs have spanners?

Answer: No, not in the sense described earlier!



Complete bipartite directed graph has $n^2/4$ edges, but no spanners!

Issue: Distance metric in directed graphs isn't symmetric

Roundtrip Distance and Spanners

Directed Graphs

Question: do directed graphs have spanners?

Answer: No, not in the sense described earlier!

Issue: Distance metric in directed graphs isn't symmetric

Roundtrip Distance and Spanners

Directed Graphs

Question: do directed graphs have spanners?

Answer: No, not in the sense described earlier!

Issue: Distance metric in directed graphs isn't symmetric

Roundtrip metric: $d_G(u \leftrightarrow v) := d_G(u,v) + d_G(v,u)$

c-Roundtrip spanner:
Subgraph H with $d_G(u \leftrightarrow v) \leq d_H(u \leftrightarrow v) \leq c \cdot d_G(u \leftrightarrow v)$

Roundtrip Distance and Spanners

Directed Graphs

Question: do directed graphs have spanners?

Answer: No, not in the sense described earlier!

Issue: Distance metric in directed graphs isn't symmetric

Roundtrip metric: $d_G(u \leftrightarrow v) := d_G(u,v) + d_G(v,u)$

Requires computing full roundtrip metric

c-Roundtrip spanner:
Subgraph H with $d_G(u \leftrightarrow v) \leq d_H(u \leftrightarrow v) \leq c \cdot d_G(u \leftrightarrow v)$

[RTZ08, CDG20] (2k-1)-roundtrip spanner exists with $O(kn^{1+1/k} \log(nW))$ edges, no efficient algorithm.

Girth Approximation: Undirected vs Directed

Girth Approximation: Undirected vs Directed

Authors	(Quality, Runtime)	Undirected/directed
[TZ05]	$(2k-1, mn^{1/k})$	Undirected
[RW12]	$(3/2, n^{5/3})$	Undirected
Trivial	(1, mn)	Directed
[PRSTV18]	$(O(k \log n), mn^{1/k})$	Directed

Our Results

Authors	(Quality, Runtime)	Undirected/directed
[TZ05]	(2k-1, $mn^{1/k}$)	Undirected
[RW12]	(3/2, $n^{5/3}$)	Undirected
Trivial	(1, mn)	Directed
[PRSTV18]	(O(k log n), $mn^{1/k}$)	Directed

Our Results

Authors	(Quality, Runtime)	Undirected/directed
[TZ05]	(2k-1, $mn^{1/k}$)	Undirected
[RW12]	(3/2, $n^{5/3}$)	Undirected
Trivial	(1, mn)	Directed
[PRSTV18]	(O(k log n), $mn^{1/k}$)	Directed
[CLRS20]	(3, $mn^{1/2}$)	Directed
[CLRS20]	(O(k log k), $mn^{1/k}$)	Directed

Our Results

Authors	(Quality, Runtime)	Undirected/directed
[TZ05]	(2k-1, $mn^{1/k}$)	Undirected
[RW12]	(3/2, $n^{5/3}$)	Undirected
Trivia	[illegible]	ected
[PRSTV1	[illegible]	rected
[CLRS20]	(3, $mn^{1/2}$)	Directed
[CLRS20]	(O(k log k), $mn^{1/k}$)	Directed

Constant factor girth approximation in subquadratic time! In fact, exponent arbitrarily close to 1.

Directed Graph Primitives Matching Undirected Graphs

Directed Graph Primitives Matching Undirected Graphs

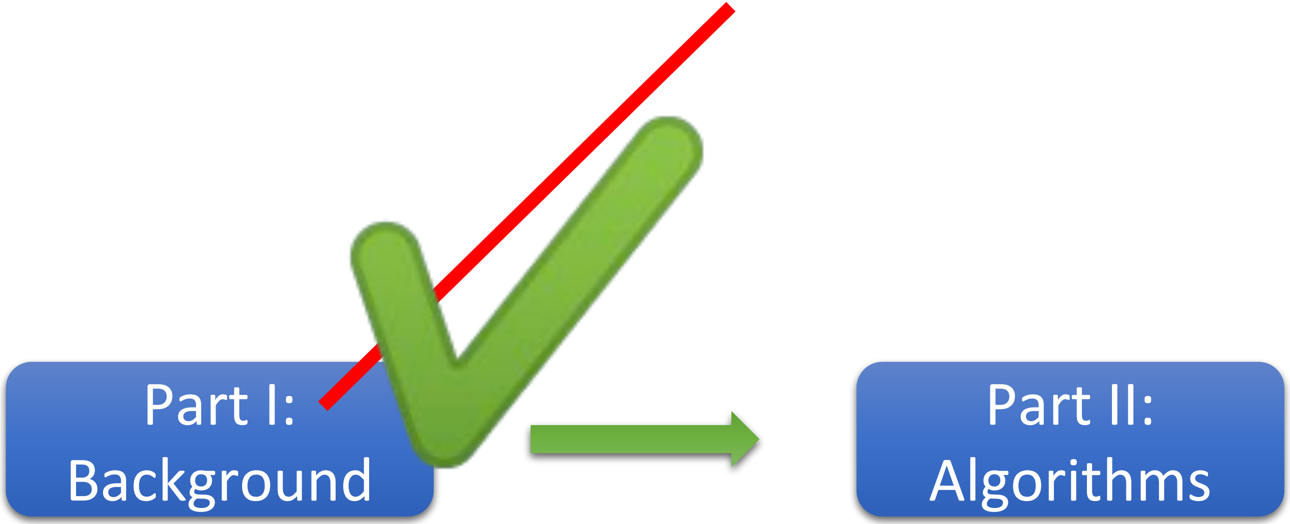
Are directed graph problems harder than undirected graph problems?

Directed Graph Primitives Matching Undirected Graphs

Are directed graph problems harder than undirected graph problems?

1. Girth approximation
2. Laplacian solving
3. Parallel reachability / shortest paths
4. Maximum flow


Part I:
Background
Part II:
Algorithms


Part I:
Background
Part II:
Algorithms

Rest of Talk

Part I: Background

Part II: Algorithms

Ball Growing for Girth Approximation and Spanners

Ball Growing for Girth Approximation and Spanners

Algorithm for (2k-1)-spanner with $O(n^{1+1/k})$ edges in unweighted undirected graph.

Ball Growing for Girth Approximation and Spanners

Algorithm for (2k-1)-spanner with $O(n^{1+1/k})$ edges in unweighted undirected graph.

Ball growing: Build a BFS/shortest-path tree around a vertex v in levels.

Cutting condition: Terminate ball growing when boundary is sparse.

Ball Growing for Girth Approximation and Spanners

Algorithm for (2k-1)-spanner with $O(n^{1+1/k})$ edges in unweighted undirected graph.

Ball growing: Build a BFS/shortest-path tree around a vertex v in levels.

Cutting condition: Terminate ball growing when boundary is sparse.

Precisely -- B(v,d) = {vertices u in V(G) within distance d of v}.

Ball Growing for Girth Approximation and Spanners

Algorithm for (2k-1)-spanner with $O(n^{1+1/k})$ edges in unweighted undirected graph.

Ball growing: Build a BFS/shortest-path tree around a vertex v in levels.

Cutting condition: Terminate ball growing when boundary is sparse.

Precisely -- B(v,d) = {vertices u in V(G) within distance d of v}.

Algorithm: pick arbitrary v, find minimal d such that $|B(v,d+1)| \le n^{1/k}|B(v,d)|$.

Add spanning tree on B(v,d+1) and delete B(v,d).

Ball Growing: Example and Analysis

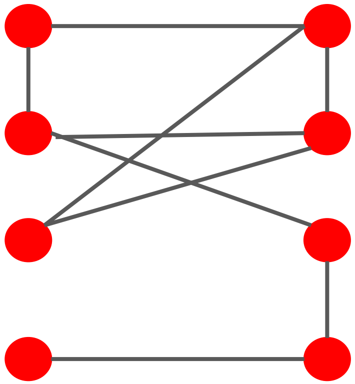
Algorithm: pick arbitrary v, find minimal d such that $|B(v,d+1)| <= n^{1/k}|B(v,d)|$.

Add spanning tree on B(v,d+1) and delete B(v,d).

Ball Growing: Example and Analysis

Algorithm: pick arbitrary v, find minimal d such that $|B(v,d+1)| <= n^{1/k}|B(v,d)|$.

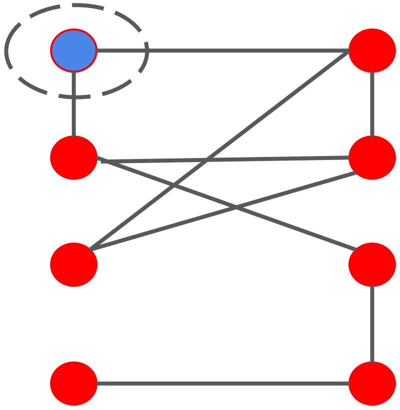
Add spanning tree on B(v,d+1) and delete B(v,d).



Ball Growing: Example and Analysis

Algorithm: pick arbitrary v, find minimal d such that $|B(v,d+1)| <= n^{1/k}|B(v,d)|$.

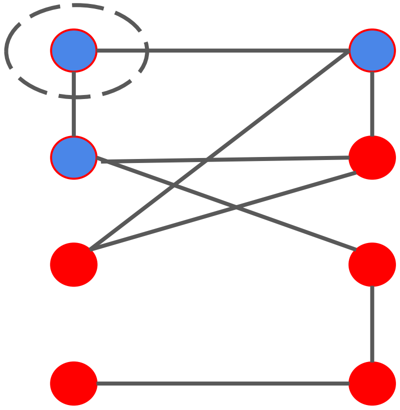
Add spanning tree on B(v,d+1) and delete B(v,d).



Ball Growing: Example and Analysis

Algorithm: pick arbitrary v, find minimal d such that |B(v,d+1)| <= $n^{1/k}$|B(v,d)|.

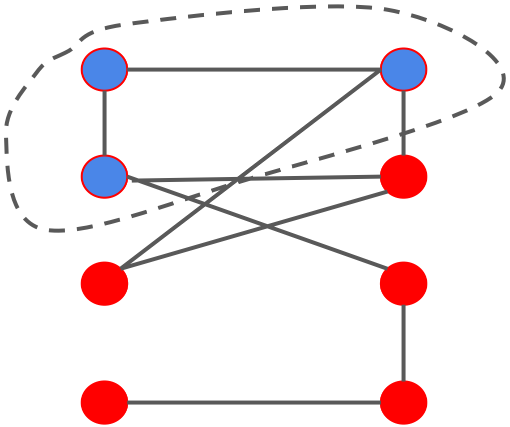
Add spanning tree on B(v,d+1) and delete B(v,d).



Ball Growing: Example and Analysis

Algorithm: pick arbitrary v, find minimal d such that $|B(v,d+1)| <= n^{1/k}|B(v,d)|$.

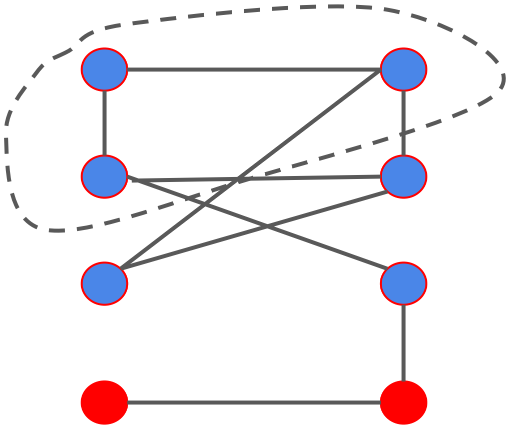
Add spanning tree on B(v,d+1) and delete B(v,d).



Ball Growing: Example and Analysis

Algorithm: pick arbitrary v, find minimal d such that $|B(v,d+1)| <= n^{1/k}|B(v,d)|$.

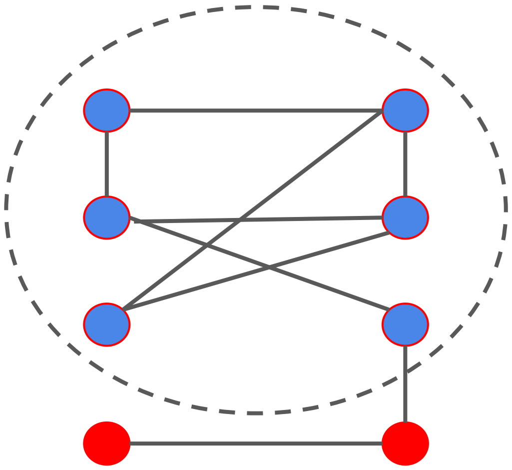
Add spanning tree on B(v,d+1) and delete B(v,d).



Ball Growing: Example and Analysis

Algorithm: pick arbitrary v, find minimal d such that $|B(v,d+1)| <= n^{1/k}|B(v,d)|$.

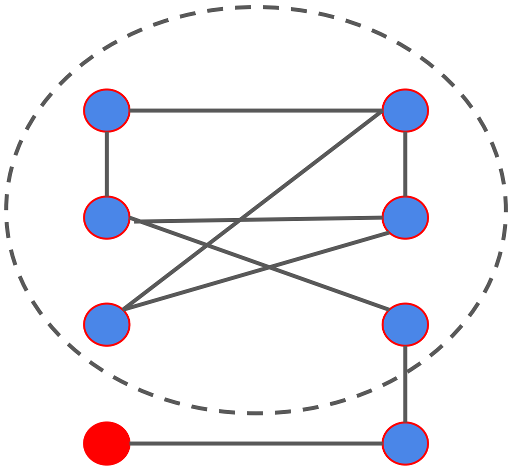
Add spanning tree on B(v,d+1) and delete B(v,d).



Ball Growing: Example and Analysis

Algorithm: pick arbitrary v, find minimal d such that $|B(v,d+1)| \le n^{1/k}|B(v,d)|$.

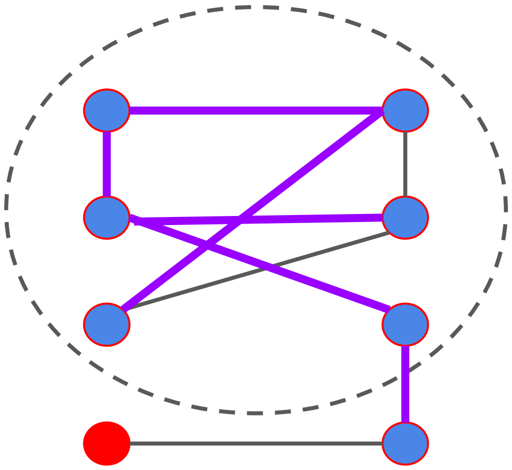
Add spanning tree on B(v,d+1) and delete B(v,d).



Ball Growing: Example and Analysis

Algorithm: pick arbitrary v, find minimal d such that $|B(v,d+1)| <= n^{1/k}|B(v,d)|$.

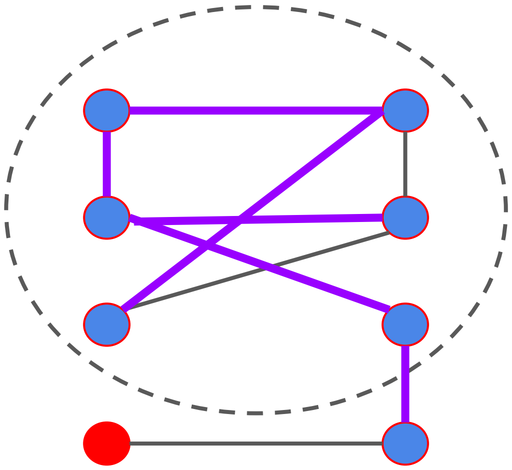
Add spanning tree on B(v,d+1) and delete B(v,d).



Ball Growing: Example and Analysis

Algorithm: pick arbitrary v, find minimal d such that $|B(v,d+1)| <= n^{1/k}|B(v,d)|$.

Add spanning tree on B(v,d+1) and delete B(v,d).

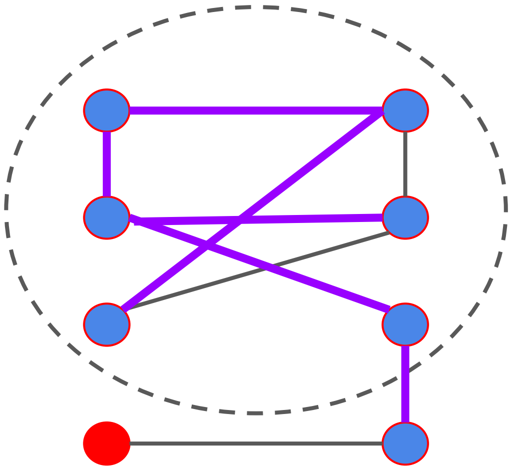


Analysis

Ball Growing: Example and Analysis

Algorithm: pick arbitrary v, find minimal d such that $|B(v,d+1)| <= n^{1/k}|B(v,d)|$.

Add spanning tree on B(v,d+1) and delete B(v,d).



Analysis

(2k-1)-spanner:
$|B(v,d+1)| <= n^{1/k}|B(v,d)|$ will be violated for some $d <= k$.

At most $n^{1+1/k}$ edges:
charge $n^{1/k}$ edges per vertex deleted by the condition $|B(v,d+1)| <= n^{1/k}|B(v,d)|$.

Undirected Girth Algorithms do not Directly Translate

Undirected Girth Algorithms do not Directly Translate

Issue 1: Growing balls in the roundtrip metric is inefficient.

Undirected Girth Algorithms do not Directly Translate

Issue 1: Growing balls in the roundtrip metric is inefficient.

Potential solution: Grow directed balls inwards and outwards.

Undirected Girth Algorithms do not Directly Translate

Issue 1: Growing balls in the roundtrip metric is inefficient.

Potential solution: Grow directed balls inwards and outwards.

Outball B(v,R): Vertices u such that d(v,u) <= R.

Undirected Girth Algorithms do not Directly Translate

Issue 1: Growing balls in the roundtrip metric is inefficient.

Potential solution: Grow directed balls inwards and outwards.

Outball B(v,R): Vertices u such that d(v,u) <= R.

Issue 2: Spanning tree on outball doesn’t give distance information.

Undirected Girth Algorithms do not Directly Translate

Issue 1: Growing balls in the roundtrip metric is inefficient.

Potential solution: Grow directed balls inwards and outwards.

Outball B(v,R): Vertices u such that d(v,u) <= R.

Issue 2: Spanning tree on outball doesn't give distance information.

Potential solution: Recurse on the outball instead of just adding a spanning tree.

Algorithm 1: Accelerated Ball Growing with Overlaps

Algorithm 1: Accelerated Ball Growing with Overlaps

Say we are searching for cycles of length R.

Algorithm 1: Accelerated Ball Growing with Overlaps

Say we are searching for cycles of length R.

For vertex v, compute outballs B(v,R), B(v,2R), …, B(v, dR).

Algorithm 1: Accelerated Ball Growing with Overlaps

Say we are searching for cycles of length R.

For vertex v, compute outballs B(v,R), B(v,2R), …, B(v, dR).

Observation: If B(v,(d+1)R) has no cycles of length R, then no vertices in B(v,dR) are involved in cycles of length R.

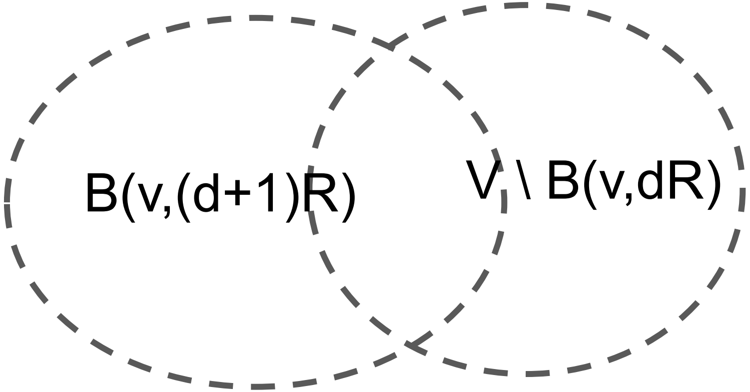
Algorithm 1: Accelerated Ball Growing with Overlaps

Say we are searching for cycles of length R.

For vertex v, compute outballs B(v,R), B(v,2R), …, B(v, dR).

Observation: If B(v,(d+1)R) has no cycles of length R, then no vertices in B(v,dR) are involved in cycles of length R.

Recurse on **overlapping** pieces B(v,(d+1)R) and V \ B(v,dR).



Algorithm 1: Accelerated Ball Growing with Overlaps

Recurse on **overlapping** pieces B(v,(d+1)R) and V \ B(v,dR).

Algorithm 1: Accelerated Ball Growing with Overlaps

Algorithm runs in $O(mn^{1/k})$ time as long as overlap is small.

Pick d to ensure that overlap is small → naively gives O(k log n) approximation.

Recurse on **overlapping** pieces B(v,(d+1)R) and V \ B(v,dR).

Algorithm 1: Accelerated Ball Growing with Overlaps

Algorithm runs in $O(mn^{1/k})$ time as long as overlap is small.

Pick d to ensure that overlap is small → naively gives O(k log n) approximation.

Observation: When B(v,dR) is small, we can allow for large boundary overlap, as the recursion will bottom out quicker.

Recurse on **overlapping** pieces B(v,(d+1)R) and V \ B(v,dR).

Algorithm 1: Accelerated Ball Growing with Overlaps

Algorithm runs in $O(mn^{1/k})$ time as long as overlap is small.

Pick d to ensure that overlap is small → naively gives O(k log n) approximation.

Observation: When B(v,dR) is small, we can allow for large boundary overlap, as the recursion will bottom out quicker.

Recurse on **overlapping** pieces B(v,(d+1)R) and V \ B(v,dR).

Theorem [CLRS20]: ***Deterministic*** **O(k loglog n) girth approximation in $O(mn^{1/k})$ time.**

Algorithm 2: Random Sampling and Distance Tests

Algorithm 2: Random Sampling and Distance Tests

Observation 1: If u is in a R-cycle with v, then d(v,u) <= R *and* for all w with d(v,w) <= R we have d(u,w) <= 2R.

Algorithm 2: Random Sampling and Distance Tests

Observation 1: If u is in a R-cycle with v, then d(v,u) <= R *and* for all w with d(v,w) <= R we have d(u,w) <= 2R.

Observation 2: If |B(v, R)| <= $n^{1/2}$, we can check all of B(v, R).

Algorithm 2: Random Sampling and Distance Tests

Observation 1: If u is in a R-cycle with v, then d(v,u) <= R *and* for all w with d(v,w) <= R we have d(u,w) <= 2R.

Observation 2: If |B(v, R)| <= $n^{1/2}$, we can check all of B(v, R).

Intuitively, use Observation 1 and random sampling to reduce to “checking” $n^{1/2}$ vertices per vertex v.

Algorithm 2: Random Sampling and Distance Tests

Algorithm 2: Random Sampling and Distance Tests

Algorithm

1. Sample M sets S_1, S_2, ... , S_M of size $O(n^{1/2}\text{polylog}(n))$.
2. Run Dijkstra for shortest paths to / from all vertices in S_i for 1 <= i <= M.

Algorithm 2: Random Sampling and Distance Tests

Algorithm

1. Sample M sets $S_1, S_2, \ldots, S_M$ of size $O(n^{1/2}\text{polylog}(n))$.
2. Run Dijkstra for shortest paths to / from all vertices in S_i for $1 <= i <= M$.
3. For $1 <= i <= M$ and each v in V(G)
4. $T_i(v) = \{s \text{ in } S_i : d(v,s) <= R \text{ and } d(s,t) <= 2R \text{ for all } t \text{ in } R_1(v),\ldots,R_{i-1}(v)\}$.
5. $R_i(v)$ is a random sample of $T_i(v)$ of size 100 log n.

Algorithm 2: Random Sampling and Distance Tests

Algorithm

1. Sample M sets $S_1, S_2, \ldots, S_M$ of size $O(n^{1/2}\text{polylog}(n))$.
2. Run Dijkstra for shortest paths to / from all vertices in S_i for $1 <= i <= M$.
3. For $1 <= i <= M$ and each v in V(G)
4. $T_i(v) = \{s \text{ in } S_i : d(v,s) <= R \text{ and } d(s,t) <= 2R \text{ for all } t \text{ in } R_1(v),\ldots,R_{i-1}(v)\}$.
5. $R_i(v)$ is a random sample of $T_i(v)$ of size 100 log n.

Intuitively, $T_i(v)$ is subset of S_i that algorithm still thinks can be in a cycle of length R with v.

Algorithm 2: Random Sampling and Distance Tests

Algorithm

1. Sample M sets S_1, S_2, ... , S_M of size $O(n^{1/2}\text{polylog}(n))$.
2. Run Dijkstra for shortest paths to / from all vertices in S_i for 1 <= i <= M.
3. For 1 <= i <= M and each v in V(G)
4. $T_i(v)$ = {s in S_i : d(v,s) <= R and d(s,t) <= 2R for all t in $R_1(v),\ldots,R_{i-1}(v)$}.
5. $R_i(v)$ is a random sample of $T_i(v)$ of size 100 log n.
6. If $|T_M(v)|$ <= 100 log n, ball grow from v.

Intuitively, $T_i(v)$ is subset of S_i that algorithm still thinks can be in a cycle of length R with v.

Algorithm 2: Random Sampling and Distance Tests

Analysis

Algorithm 2: Random Sampling and Distance Tests

Analysis

Dijkstra to/from S_i take $O(mn^{1/2}$ polylog(n)) total time.

Can show $|T_i(v)| <= .9\ |T_{i-1}(v)|$ with high probability or find cycle of length 4R.

If $|T_i(v)| <= O(\log n)$ then ball growing from v only visits $O(n^{1/2})$ vertices.

Algorithm 2: Random Sampling and Distance Tests

Analysis

Dijkstra to/from S_i take $O(mn^{1/2}$ polylog(n)) total time.

Can show $|T_i(v)| <= .9\ |T_{i-1}(v)|$ with high probability or find cycle of length 4R.

If $|T_i(v)| <= O(\log n)$ then ball growing from v only visits $O(n^{1/2})$ vertices.

Result: With high probability, $O(mn^{1/2})$ runtime and 4-approximation.

Get 3-approximation by being more careful.

Algorithm 3: Combination of Algo 1 and 2

Algorithm 3: Combination of Algo 1 and 2

Step 1: Sample $O(n^{1/k} \text{polylog}(n))$ random vertices, run Dijkstra from them.

Algorithm 2 reduces “important vertices to check” to $O(n^{1-1/k})$ per vertex.

Algorithm 3: Combination of Algo 1 and 2

Step 1: Sample $O(n^{1/k}$ polylog(n)) random vertices, run Dijkstra from them.

Algorithm 2 reduces "important vertices to check" to $O(n^{1-1/k})$ per vertex.

Step 2: Run accelerated ball growing (Algorithm 1).

Turns out only need to take $d = O(k \log k)$ levels now!

Algorithm 3: Combination of Algo 1 and 2

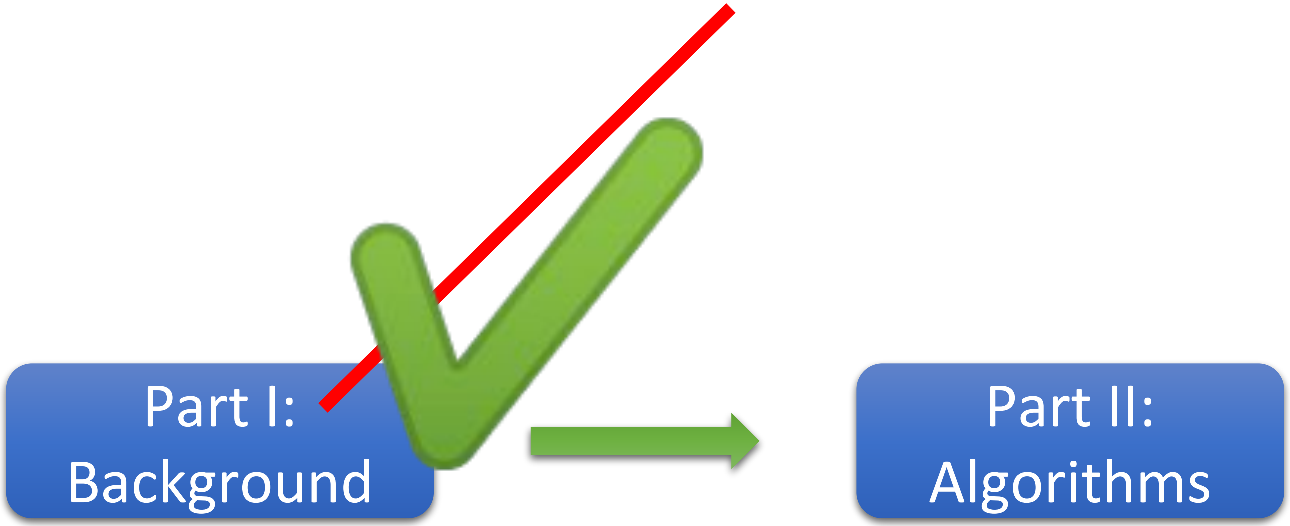
Step 1: Sample $O(n^{1/k}$ polylog(n)) random vertices, run Dijkstra from them.

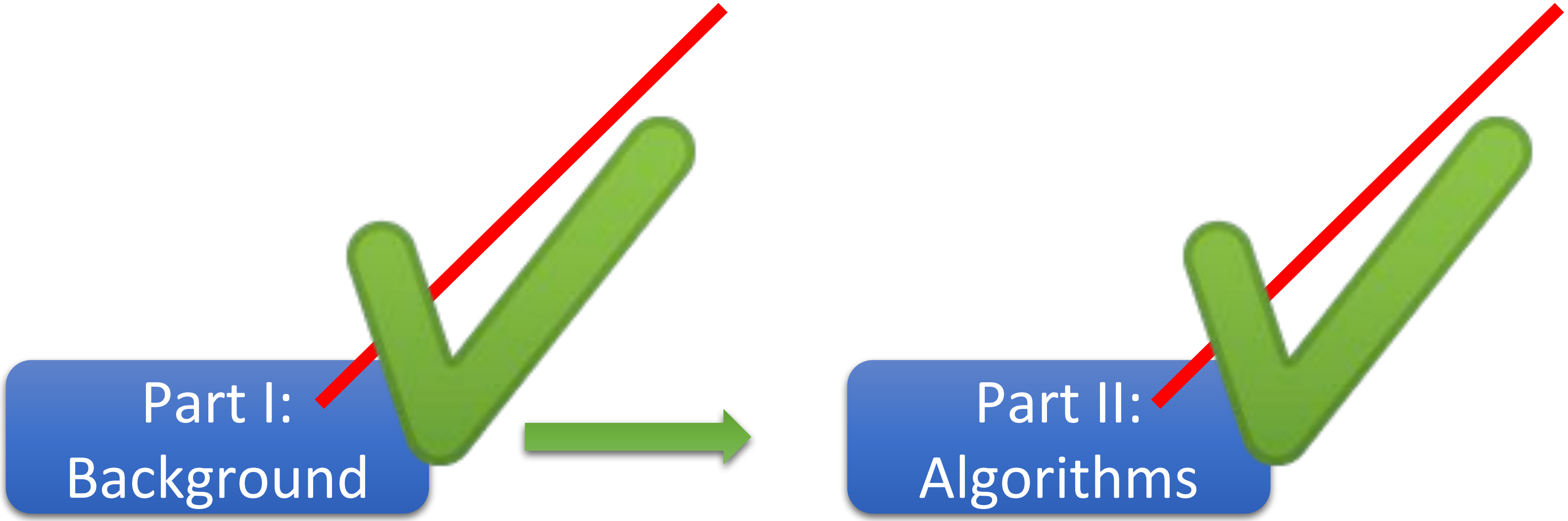
Algorithm 2 reduces "important vertices to check" to $O(n^{1-1/k})$ per vertex.

Step 2: Run accelerated ball growing (Algorithm 1).

Turns out only need to take $d = O(k \log k)$ levels now!

Result: $O(k \log k)$ girth approximation in $mn^{1/k}$ time.


Part I:
Background
Part II:
Algorithms


Part I:
Background
Part II:
Algorithms

Future Directions / Problems

Future Directions / Problems

Main Open Question: Get 2k girth approximation in $O(mn^{1/k})$ time.

Future Directions / Problems

Main Open Question: Get 2k girth approximation in $O(mn^{1/k})$ time.

Approach / Problem: (Dis)prove that for a directed graph G with m edges and no cycle of length $O(\varepsilon^{-1}\log n)$ we can remove εm edges from G and get a DAG.

Future Directions / Problems

Main Open Question: Get 2k girth approximation in $O(mn^{1/k})$ time.

Approach / Problem: (Dis)prove that for a directed graph G with m edges and no cycle of length $O(\varepsilon^{-1}\log n)$ we can remove εm edges from G and get a DAG.

[Seymour95] -- True for $O(\varepsilon^{-1}\log n \log\log n)$.

Future Directions / Problems

Main Open Question: Get 2k girth approximation in $O(mn^{1/k})$ time.

Approach / Problem: (Dis)prove that for a directed graph G with m edges and no cycle of length $O(\varepsilon^{-1}\log n)$ we can remove εm edges from G and get a DAG.

[Seymour95] -- True for $O(\varepsilon^{-1}\log n \log\log n)$.

Additional potential applications in approximation algorithm for minimum feedback arc set, property testing, etc.

Future Directions / Problems

Main Open Question: Get 2k girth approximation in $O(mn^{1/k})$ time.

Approach / Problem: (Dis)prove that for a directed graph G with m edges and no cycle of length $O(\varepsilon^{-1}\log n)$ we can remove εm edges from G and get a DAG.

[Seymour95] -- True for $O(\varepsilon^{-1}\log n \log\log n)$.

Additional potential applications in approximation algorithm for minimum feedback arc set, property testing, etc.

Recent Work [DW20]: 2-approximation in subquadratic time. Significantly improved constants in the O(k log k) approximation, eg. 4-approximation in $mn^{.414}$.

Constant Girth Approximation for Directed Graphs in Subquadratic Time

arXiv: 1907.10779

The End

Questions?

Yang P. Liu

Contact Info:

- email: *yangpliu@stanford.edu*
- website: *yangpliu.github.io*

Constant Girth Approximation for Directed Graphs in Subquadratic Time

arXiv: 1907.10779

The End

Questions?

Yang P. Liu

Contact Info:

- email: *yangpliu@stanford.edu*
- website: *yangpliu.github.io*