Dynamic algorithms for k-center on graphs

Sebastian Forster, né Krinninger

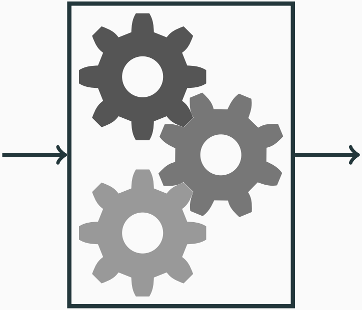
University of Salzburg

@Simons Institute (September 19, 2023)

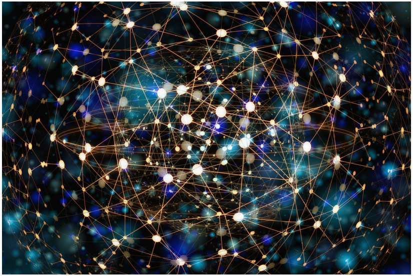
Joint work with Emilio Cruciani, Gramoz Goranci, Yasamin Nazari, and Antonis Skarlatos

Supported by the Austrian Science Fund (FWF): P 32863-N. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 947702).

Dynamic Algorithms

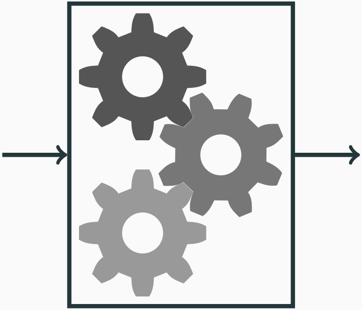


Network Science

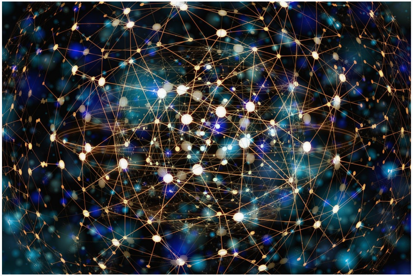


Dynamic Graph Clustering

Dynamic Algorithms



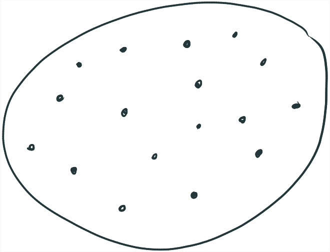
Network Science



Prior algorithms on dynamic clustering not tailored to graphs!

k-Center Problem

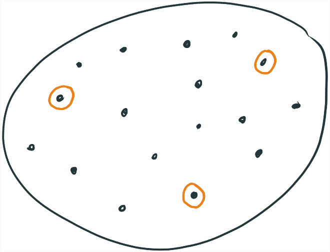
Given a metric space, select k points as set of centers C such that the maximum distance $d(C, v)$ of any node v to its closest center is minimized.



k-Center Clustering

k-Center Problem

Given a metric space, select k points as set of centers C such that the maximum distance $d(C, v)$ of any node v to its closest center is minimized.

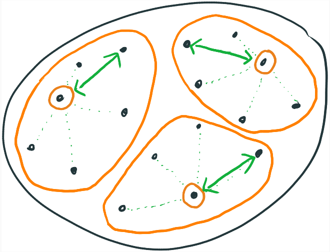


k-Center Clustering

k-Center Problem

Given a metric space, select k points as set of centers C such that the maximum distance $d(C, v)$ of any node v to its closest center is minimized.

- Assigning each point to its closest center induces a partition into clusters
- Radius of a cluster: Maximum distance of the center to the nodes in its cluster

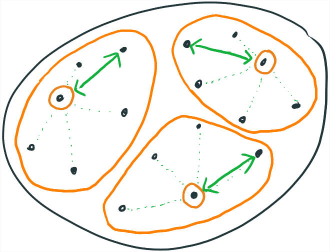


k-Center Clustering

k-Center Problem

Given a metric space, select k points as set of centers C such that the maximum distance $d(C, v)$ of any node v to its closest center is minimized.

- Assigning each point to its closest center induces a partition into clusters
- Radius of a cluster: Maximum distance of the center to the nodes in its cluster
- Problem is NP-hard to approximate within a factor of $2 - \epsilon$



Metric Spaces and Graphs

Definition (Metric on Point Set)

1. **Non-Negativity:** $d(x, y) \geq 0$
2. **Separation:** $d(x, y) = 0$ if and only if $x = y$
3. **Symmetry:** $d(x, y) = d(y, x)$
4. **Triangle inequality:** $d(x, z) \leq d(x, y) + d(y, z)$

Metric Spaces and Graphs

Definition (Metric on Point Set)

1. **Non-Negativity:** $d(x, y) \geq 0$
2. **Separation:** $d(x, y) = 0$ if and only if $x = y$
3. **Symmetry:** $d(x, y) = d(y, x)$
4. **Triangle inequality:** $d(x, z) \leq d(x, y) + d(y, z)$

Pairwise shortest path distances of an undirected graph induce a metric with nodes as the point set

Metric Spaces and Graphs

Definition (Metric on Point Set)

1. **Non-Negativity:** $d(x, y) \geq 0$
2. **Separation:** $d(x, y) = 0$ if and only if $x = y$
3. **Symmetry:** $d(x, y) = d(y, x)$
4. **Triangle inequality:** $d(x, z) \leq d(x, y) + d(y, z)$

Pairwise shortest path distances of an undirected graph induce a metric with nodes as the point set

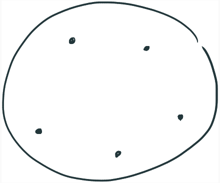
Question

Are there efficient dynamic constant-factor approximation algorithms for k-center if the metric is induced by a dynamically changing undirected graph?

Dynamic Model

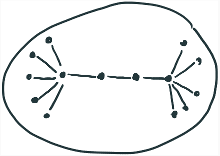
Dynamic point sets:

- Point insertions and deletions



Dynamic graphs:

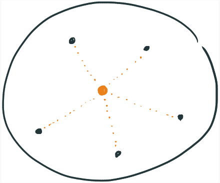
- Edge insertions and deletions



Dynamic Model

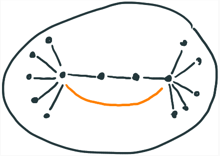
Dynamic point sets:

- Point insertions and deletions



Dynamic graphs:

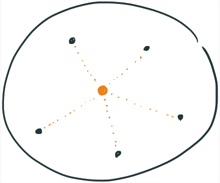
- Edge insertions and deletions



Dynamic Model

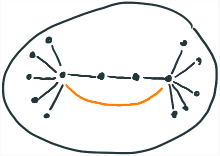
Dynamic point sets:

- Point insertions and deletions
- Query access to metric



Dynamic graphs:

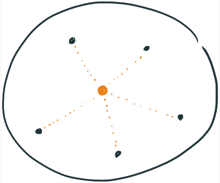
- Edge insertions and deletions
- Distances not given for free



Dynamic Model

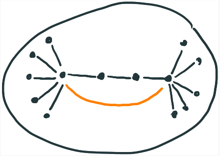
Dynamic point sets:

- Point insertions and deletions
- Query access to metric
- Metric is extended/reduced



Dynamic graphs:

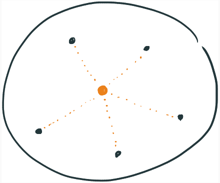
- Edge insertions and deletions
- Distances not given for free
- Metric is shrinking/expanding



Dynamic Model

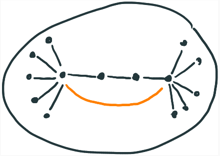
Dynamic point sets:

- Point insertions and deletions
- Query access to metric
- Metric is extended/reduced



Dynamic graphs:

- Edge insertions and deletions
- Distances not given for free
- Metric is shrinking/expanding



Conclusion

Cannot use results for dynamic point sets in a black-box manner for dynamic graph model

Related Work

Static algorithms:

- Classic 2-approximation algorithms [Gonzalez '85] [Hochbaum, Shmoys '85]
  On graphs with n nodes and m edges: $\tilde{O}(km)$ time
- State of the art on graphs: $\tilde{O}(m)$ time (randomized) [Thorup '01] [Abboud et al. '23]

Related Work

Static algorithms:

- Classic 2-approximation algorithms [Gonzalez '85] [Hochbaum, Shmoys '85]
  On graphs with n nodes and m edges: $\tilde{O}(km)$ time
- State of the art on graphs: $\tilde{O}(m)$ time (randomized) [Thorup '01] [Abboud et al. '23]

Dynamic point sets:

- $\tilde{O}(k^2)$ update time [Chan, Gourqin, Sozio '18]
- $\tilde{O}(k)$ update time [Bateni et al. '23]
- Special cases: [Schmidt, Sohler '19] [Goranci et al. '21]
- Consistent k-center [Lattanzi and Vassilvitskii '12] [Fichtenberger et al. '21] [Łącki et al. '23]

Related Work

Static algorithms:

- Classic 2-approximation algorithms [Gonzalez '85] [Hochbaum, Shmoys '85]
  On graphs with n nodes and m edges: $\tilde{O}(km)$ time
- State of the art on graphs: $\tilde{O}(m)$ time (randomized) [Thorup '01] [Abboud et al. '23]

Dynamic point sets:

- $\tilde{O}(k^2)$ update time [Chan, Gourqin, Sozio '18]
- $\tilde{O}(k)$ update time [Bateni et al. '23]
- Special cases: [Schmidt, Sohler '19] [Goranci et al. '21]
- Consistent k-center [Lattanzi and Vassilvitskii '12] [Fichtenberger et al. '21] [Łącki et al. '23]

Natural goal: Update-time overhead of $\tilde{O}(k)$ compared to dynamic approximate SSSP

Our Results I: Fully Dynamic

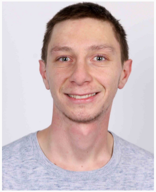
Theorem (Cruciani, F, Goranci, Nazari, Skarlatos ’23)

There is a fully dynamic $(2+\epsilon)$-approximate k-center algorithm with worst-case update time

- $O(kn^{1.529}\epsilon^{-2})$ *in unweighted graphs*
- $O(kn^{1.823}\epsilon^{-2})$ *in weighted graphs*

that is correct against an adaptive adversary.



Our Results I: Fully Dynamic

Theorem (Cruciani, F, Goranci, Nazari, Skarlatos ’23)

There is a fully dynamic $(2+\epsilon)$-approximate k-center algorithm with worst-case update time

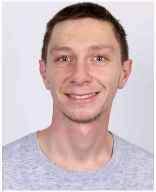
- $O(kn^{1.529}\epsilon^{-2})$ *in unweighted graphs*
- $O(kn^{1.823}\epsilon^{-2})$ *in weighted graphs*

that is correct against an adaptive adversary.

Update time for fully dynamic $(1+\epsilon)$-approximate SSSP:

- $O(n^{1.529}\epsilon^{-2})$ (unweighted) [v. d. Brand, **F**, Nazari ’22]
- $O(n^{1.823}\epsilon^{-2})$ (weighted) [v. d. Brand, Nanongkai ’19]



Our Results II: Partially Dynamic

Theorem (Cruciani, F, Goranci, Nazari, Skarlatos '23)

There is a deterministic decremental $(2+\epsilon)$-approximate k-center algorithm with amortized update time $kn^{o(1)}$ (over a sequence of $\Theta(m)$ updates).

(in this talk: constant ϵ, polynomially bounded integer edge weights)

Update time for decremental $(1+\epsilon)$-approximate SSSP: $n^{o(1)}$

Our Results II: Partially Dynamic

Theorem (Cruciani, F, Goranci, Nazari, Skarlatos '23)

There is a deterministic decremental $(2+\epsilon)$-approximate k-center algorithm with amortized update time $kn^{o(1)}$ (over a sequence of $\Theta(m)$ updates).

(in this talk: constant ϵ, polynomially bounded integer edge weights)

Update time for decremental $(1+\epsilon)$-approximate SSSP: $n^{o(1)}$

Theorem (Cruciani, F, Goranci, Nazari, Skarlatos '23)

There is a randomized incremental $(4+\epsilon)$-approximate k-center algorithm with amortized update time $kn^{o(1)}$ that is correct against an oblivious adversary.

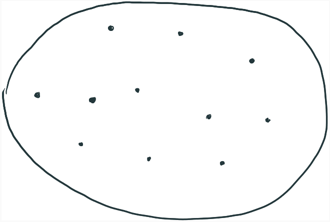
Update time for incremental $(1+\epsilon)$-approximate SSSP: $n^{o(1)}$

Warm-Up: Fully Dynamic Algorithm

Reminder: Gonzalez's Algorithm

Gonzalez's Algorithm [Gonzalez '85]

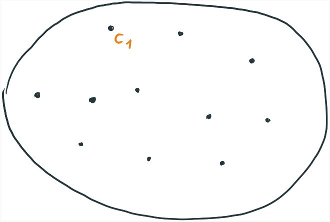
1. Initialize $C = \{v\}$ with arbitrary first center
2. While $|C| < k$, add node v maximizing $d(C, v)$ to C



Reminder: Gonzalez's Algorithm

Gonzalez's Algorithm [Gonzalez '85]

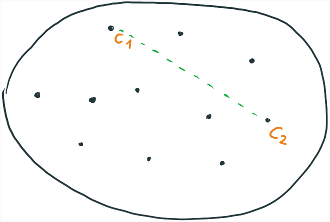
1. Initialize $C = \{v\}$ with arbitrary first center
2. While $|C| < k$, add node v maximizing $d(C, v)$ to C



Reminder: Gonzalez's Algorithm

Gonzalez's Algorithm [Gonzalez '85]

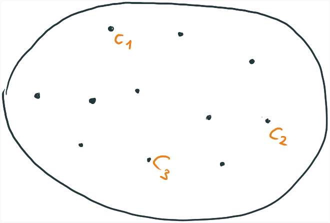
1. Initialize $C = \{v\}$ with arbitrary first center
2. While $|C| < k$, add node v maximizing $d(C, v)$ to C



Reminder: Gonzalez's Algorithm

Gonzalez's Algorithm [Gonzalez '85]

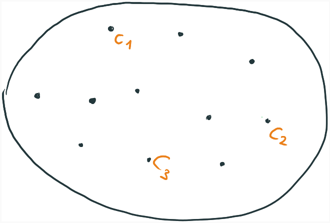
1. Initialize $C = \{v\}$ with arbitrary first center
2. While $|C| < k$, add node v maximizing $d(C, v)$ to C



Reminder: Gonzalez's Algorithm

Gonzalez's Algorithm [Gonzalez '85]

1. Initialize $C = \{v\}$ with arbitrary first center
2. While $|C| < k$, add node v maximizing $d(C, v)$ to C

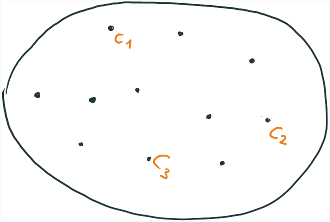


This gives a 2-approximation

Reminder: Gonzalez's Algorithm

Gonzalez's Algorithm [Gonzalez '85]

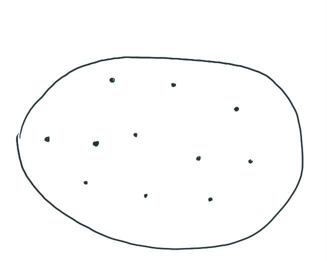
1. Initialize $C = \{v\}$ with arbitrary first center
2. While $|C| < k$, add node v maximizing $d(C, v)$ to C



This gives a 2-approximation

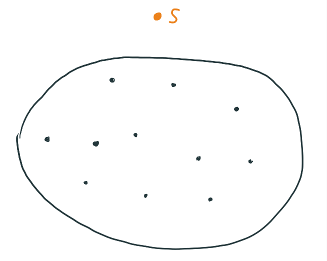
If $d(C, v)$ is within factor $1 + \epsilon$ of maximum, this gives $(2 + \epsilon)$-approximation

Reduction to SSSP: Simulating Gonzalez's Algorithm



Reduction to SSSP: Simulating Gonzalez's Algorithm

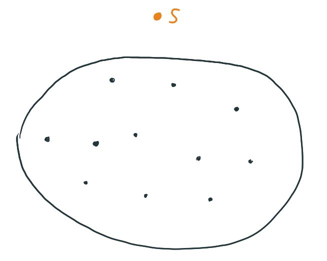
- Add artificial "super-source" s
- Maintain $(1+\epsilon)$-approximate single-source distances from s with a fully dynamic algorithm



Reduction to SSSP: Simulating Gonzalez's Algorithm

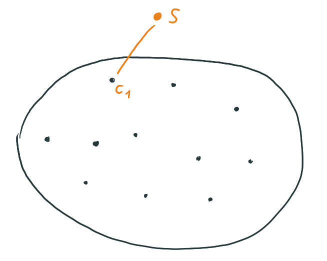
- Add artificial "super-source" s
- Maintain $(1+\epsilon)$-approximate single-source distances from s with a fully dynamic algorithm

- After every update to graph:
  - Forward update to SSSP data structure



Reduction to SSSP: Simulating Gonzalez's Algorithm

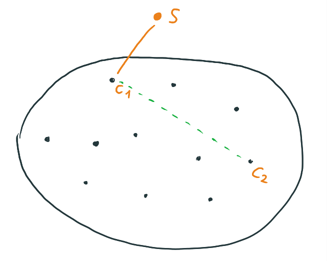
- Add artificial "super-source" s
- Maintain $(1+\epsilon)$-approximate single-source distances from s with a fully dynamic algorithm



- After every update to graph:
  - Forward update to SSSP data structure
  - Initialize $C = \{v\}$ with arbitrary first center and connect it to s

Reduction to SSSP: Simulating Gonzalez's Algorithm

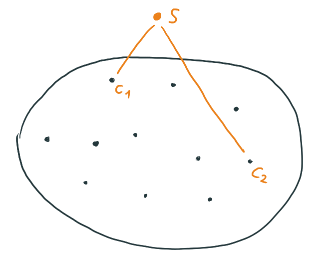
- Add artificial "super-source" s
- Maintain $(1+\epsilon)$-approximate single-source distances from s with a fully dynamic algorithm



- After every update to graph:
  - Forward update to SSSP data structure
  - Initialize $C = \{v\}$ with arbitrary first center and connect it to s
  - While $|C| < k$, add node v maximizing $d(s, v)$ to C and connect it to s

Reduction to SSSP: Simulating Gonzalez's Algorithm

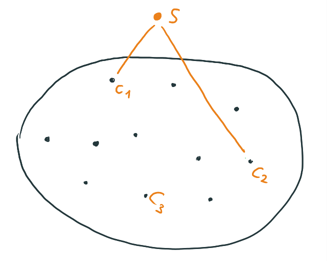
- Add artificial "super-source" s
- Maintain $(1+\epsilon)$-approximate single-source distances from s with a fully dynamic algorithm



- After every update to graph:
  - Forward update to SSSP data structure
  - Initialize $C = \{v\}$ with arbitrary first center and connect it to s
  - While $|C| < k$, add node v maximizing $d(s, v)$ to C and connect it to s

Reduction to SSSP: Simulating Gonzalez's Algorithm

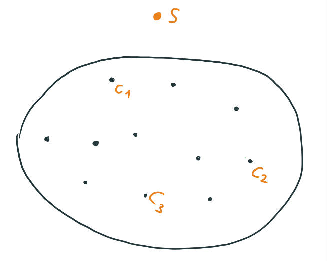
- Add artificial "super-source" s
- Maintain $(1+\epsilon)$-approximate single-source distances from s with a fully dynamic algorithm



- After every update to graph:
  - Forward update to SSSP data structure
  - Initialize $C = \{v\}$ with arbitrary first center and connect it to s
  - While $|C| < k$, add node v maximizing $d(s, v)$ to C and connect it to s

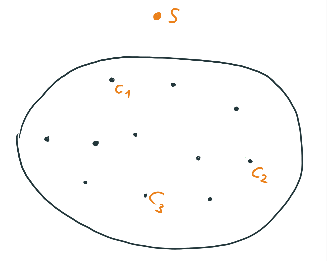
Reduction to SSSP: Simulating Gonzalez's Algorithm

- Add artificial "super-source" s
- Maintain $(1+\epsilon)$-approximate single-source distances from s with a fully dynamic algorithm **with algorithm working against adaptive adversary**
- After every update to graph:
    - Forward update to SSSP data structure
    - Initialize $C = \{v\}$ with arbitrary first center and connect it to s
    - While $|C| < k$, add node v maximizing $d(s, v)$ to C and connect it to s



Reduction to SSSP: Simulating Gonzalez's Algorithm

- Add artificial "super-source" s
- Maintain $(1+\epsilon)$-approximate single-source distances from s with a fully dynamic algorithm **with algorithm working against adaptive adversary**
- After every update to graph:
  - Forward update to SSSP data structure
  - Initialize $C = \{v\}$ with arbitrary first center and connect it to s
  - While $|C| < k$, add node v maximizing $d(s, v)$ to C and connect it to s



Update Time: $O(k \cdot U_{\mathrm{SSSP}}(n))$

Partially Dynamic Algorithms

R-Independent Sets

Definition (R-Independent Set)

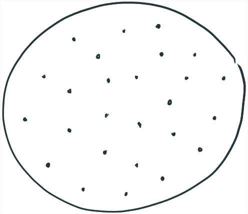
- $C \subseteq V$ such that $d(u, v) > R$ for all $u, v \in C$
- *maximal* if C cannot be extended without violating the property

R-Independent Sets

Definition (R-Independent Set)

- $C \subseteq V$ such that $d(u, v) > R$ for all $u, v \in C$
- *maximal* if C cannot be extended without violating the property

Greedy algorithm:

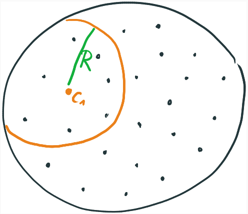


R-Independent Sets

Definition (R-Independent Set)

- $C \subseteq V$ such that $d(u, v) > R$ for all $u, v \in C$
- *maximal* if C cannot be extended without violating the property

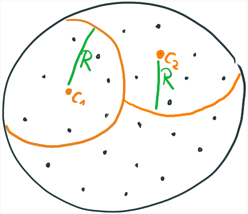
Greedy algorithm:



Definition (R-Independent Set)

- $C \subseteq V$ such that $d(u, v) > R$ for all $u, v \in C$
- *maximal* if C cannot be extended without violating the property

Greedy algorithm:

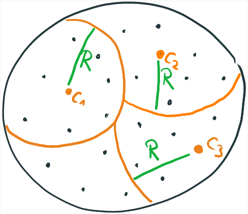


R-Independent Sets

Definition (R-Independent Set)

- $C \subseteq V$ such that $d(u, v) > R$ for all $u, v \in C$
- *maximal* if C cannot be extended without violating the property

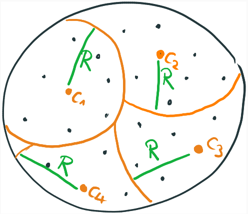
Greedy algorithm:



Definition (R-Independent Set)

- $C \subseteq V$ such that $d(u, v) > R$ for all $u, v \in C$
- *maximal* if C cannot be extended without violating the property

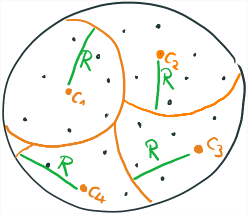
Greedy algorithm:



Definition (R-Independent Set)

- $C \subseteq V$ such that $d(u, v) > R$ for all $u, v \in C$
- *maximal* if C cannot be extended without violating the property

Greedy algorithm:



"k-bounded maximal R-Independent set C":

- $|C| < k$ and $|C|$ is maximal or
- $|C| = k$

k-Center Based on R-Independent Sets [Hochbaum, Shmoys '85]

Goal: Find smallest value of R such that maximal R-independent set has size $\leq k$

k-Center Based on R-Independent Sets [Hochbaum, Shmoys '85]

Goal: Find smallest value of R such that maximal R-independent set has size $\leq k$

Algorithm:

- Obtain "guess" of R by binary search or taking powers of $1 + \epsilon$
- Compute maximal R-independent set C
- If $|C| > k$, increase R
- If $|C| \leq k$, decrease R

k-Center Based on R-Independent Sets [Hochbaum, Shmoys '85]

Goal: Find smallest value of R such that maximal R-independent set has size $\leq k$

Algorithm:

- Obtain "guess" of R by binary search or taking powers of $1 + \epsilon$
- Compute maximal R-independent set C
- If $|C| > k$, increase R
- If $|C| \leq k$, decrease R

Efficiency: Compute k-bounded maximal R-independent and check if it is indeed maximal

k-Center Based on R-Independent Sets [Hochbaum, Shmoys '85]

Goal: Find smallest value of R such that maximal R-independent set has size $\leq k$

Algorithm:

- Obtain "guess" of R by binary search or taking powers of $1 + \epsilon$
- Compute maximal R-independent set C
- If $|C| > k$, increase R
- If $|C| \leq k$, decrease R

Efficiency: Compute k-bounded maximal R-independent and check if it is indeed maximal

Lemma ([Hochbaum, Shmoys '85])

If $R \geq 2OPT_k$, then every maximal R-independent set has size $\leq k$

k-Center Based on R-Independent Sets [Hochbaum, Shmoys '85]

Goal: Find smallest value of R such that maximal R-independent set has size $\leq k$

Algorithm:

- Obtain "guess" of R by binary search or taking powers of $1 + \epsilon$
- Compute maximal R-independent set C
- If $|C| > k$, increase R
- If $|C| \leq k$, decrease R

Efficiency: Compute k-bounded maximal R-independent and check if it is indeed maximal

Lemma ([Hochbaum, Shmoys '85])

If $R \geq 2OPT_k$, then every maximal R-independent set has size $\leq k$

Again: $(1 + \epsilon)$-approximate distances lead to $(2 + \epsilon)$-approximation

Decremental Algorithm

Observations

- Distances are non-decreasing over time
- R-independent set will always continue being an R-independent set

Decremental Algorithm

Observations

- Distances are non-decreasing over time
- R-independent set will always continue being an R-independent set

Algorithm

→ Increase guess for R over time

→ Increase number of centers for current guess of R over time

Decremental Algorithm

Observations

- Distances are non-decreasing over time
- R-independent set will always continue being an R-independent set

Algorithm

→ Increase guess for R over time

→ Increase number of centers for current guess of R over time

→ Maintain approximate decremental SSSP from super-source connected to every center

Decremental Algorithm

Observations

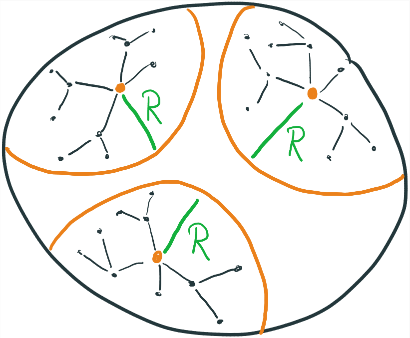
- Distances are non-decreasing over time
- R-independent set will always continue being an R-independent set

Algorithm

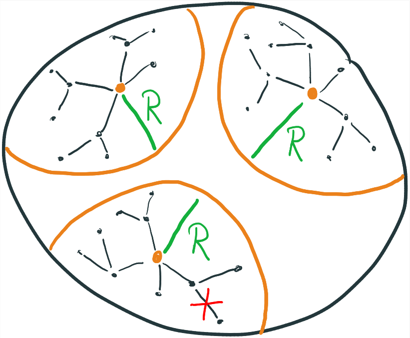
→ Increase guess for R over time
→ Increase number of centers for current guess of R over time
→ Maintain approximate decremental SSSP from super-source connected to every center

- After each deletion:
  - Forward update to SSSP data structure
  - If there is a node with $d(C, v) > (1 + \epsilon)R$:
    - If $|C| = k$, then $R \leftarrow (1 + \epsilon)R$
    - Otherwise: Add v to C and restart decremental SSSP

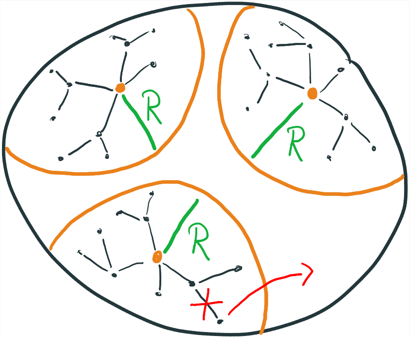
Example and Running Time



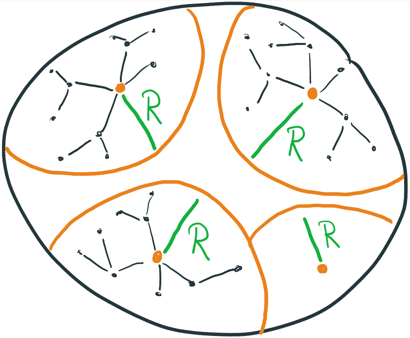
Example and Running Time



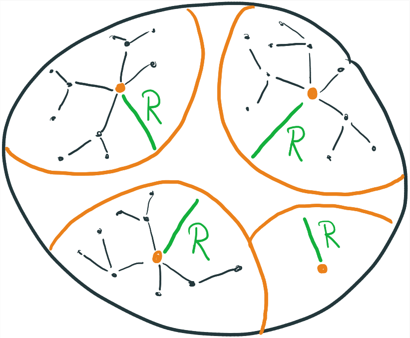
Example and Running Time



Example and Running Time



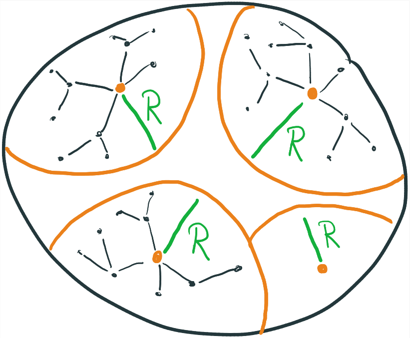
Example and Running Time



Total update time: $O(k \log_{1+\epsilon}(n\Lambda)) \times T_{\mathrm{SSSP}}(m)$

- For each guess of R, $|C|$ increases at most k times
- Decremental SSSP is restarted $O(k \log_{1+\epsilon}(n\Lambda))$ times, where Λ is the aspect ratio of the graph

Example and Running Time



Total update time: $O(k \log_{1+\epsilon}(n\Lambda)) \times T_{\mathrm{SSSP}}(m)$

- For each guess of R, $|C|$ increases at most k times
- Decremental SSSP is restarted $O(k \log_{1+\epsilon}(n\Lambda))$ times, where Λ is the aspect ratio of the graph
- $T_{\mathrm{SSSP}}(m) = m^{1+o(1)}$ [Henzinger, **K**, Nanongkai '14], also deterministically [Bernstein, Probst Gutenberg, Saranurak '21]

Incremental Algorithm

Observations

- Distances are non-increasing over time

Incremental Algorithm

Observations

- Distances are non-increasing over time
- **BUT:** R-independent will not necessarily continue being an R-independent set

Incremental Algorithm

Observations

- Distances are non-increasing over time
- **BUT:** R-independent will not necessarily continue being an R-independent set

If insertion leads to conflict in R-independent set C, we need to update/recompute C because we want $|C| \leq k$

Incremental Algorithm

Observations

- Distances are non-increasing over time
- **BUT:** R-independent will not necessarily continue being an R-independent set

If insertion leads to conflict in R-independent set C, we need to update/recompute C because we want $|C| \leq k$

Efficiency Problem:

- Maintain approximate SSSP from every node in C
- Every change to C is expensive!
  $\rightarrow$ Total update time time:
  (#nodes ever contained in C) $\times\, m^{1+o(1)}$

Incremental Algorithm

Observations

- Distances are non-increasing over time
- **BUT:** R-independent will not necessarily continue being an R-independent set

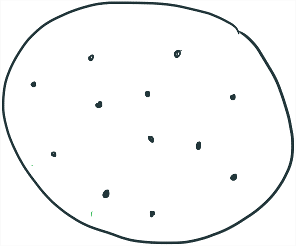
If insertion leads to conflict in R-independent set C, we need to update/recompute C because we want $|C| \leq k$

Efficiency Problem:

- Maintain approximate SSSP from every node in C
- Every change to C is expensive!
  $\rightarrow$ Total update time time:
  (#nodes ever contained in C) $\times\, m^{1+o(1)}$
- **Goal:** Maintain R-independent sets with low **total** recourse

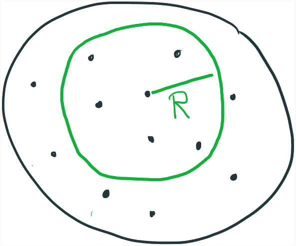
MIS Abstraction [Bateni et al. '23]

R-**neighborhood graph** G_R: edge (x, y) if $d(x, y) \leq R$



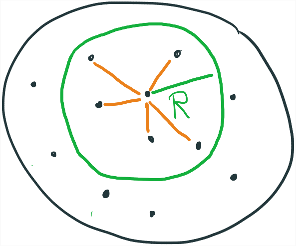
MIS Abstraction [Bateni et al. '23]

R-**neighborhood graph** G_R: edge (x, y) if $d(x, y) \le R$



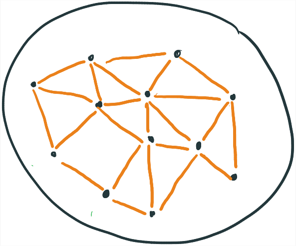
MIS Abstraction [Bateni et al. '23]

R-**neighborhood graph** G_R: edge (x, y) if $d(x, y) \leq R$



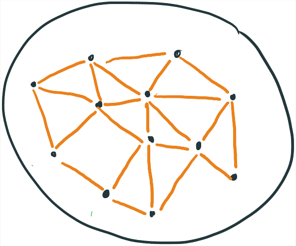
MIS Abstraction [Bateni et al. '23]

R-**neighborhood graph** G_R: edge (x, y) if $d(x, y) \leq R$



MIS Abstraction [Bateni et al. '23]

R-**neighborhood graph** G_R: edge (x, y) if $d(x, y) \le R$



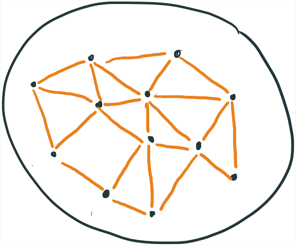
Independent set in G_R

=

R-independent set in original graph

MIS Abstraction [Bateni et al. '23]

R-**neighborhood graph** G_R: edge (x, y) if $d(x, y) \leq R$



Independent set in G_R

=

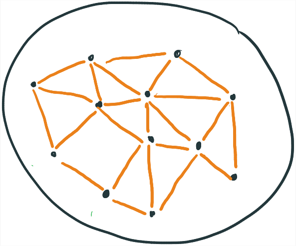
R-independent set in original graph

Remarks:

- Neighborhood graph essentially an analysis tool, only constructed partially

MIS Abstraction [Bateni et al. '23]

R-**neighborhood graph** G_R: edge (x, y) if $d(x, y) \leq R$



Independent set in G_R

=

R-independent set in original graph

Remarks:

- Neighborhood graph essentially an analysis tool, only constructed partially
- "k-bounded maximal independent set"

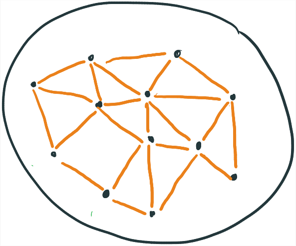
MIS Abstraction [Bateni et al. '23]

R-**neighborhood graph** G_R: edge (x, y) if $d(x, y) \leq R$

Independent set in G_R

=

R-independent set in original graph



Remarks:

- Neighborhood graph essentially an analysis tool, only constructed partially
- "k-bounded maximal independent set"
- Clean formal definition for algorithmically defined approximate distances is tricky (but also not necessary)

Dominating Sets to the Rescue

Goal:

- Maintain k-bounded MIS in neighborhood graph G_R with total recourse poly(k)

Dominating Sets to the Rescue

Goal:

- Maintain k-bounded MIS in neighborhood graph G_R with total recourse poly(k)
- Since input graph is incremental, neighborhood graph is incremental with $O(n^2)$ insertions

Dominating Sets to the Rescue

Goal:

- Maintain k-bounded MIS in neighborhood graph G_R with total recourse poly(k)
- Since input graph is incremental, neighborhood graph is incremental with $O(n^2)$ insertions
- Known dynamic MIS algorithms only provide $O(1)$ (expected) recourse *per update* [Behnezhad et al. '19] [Chechik, Zhang '19]

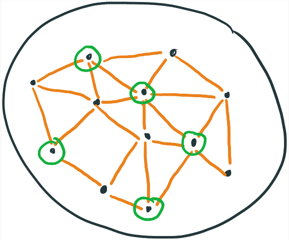
Dominating Sets to the Rescue

Goal:

- Maintain k-bounded MIS in neighborhood graph G_R with total recourse poly(k)
- Since input graph is incremental, neighborhood graph is incremental with $O(n^2)$ insertions
- Known dynamic MIS algorithms only provide $O(1)$ (expected) recourse *per update* [Behnezhad et al. '19] [Chechik, Zhang '19]

Idea:

- Maintain dominating set S on G_R with total recourse $\tilde{O}(k)$



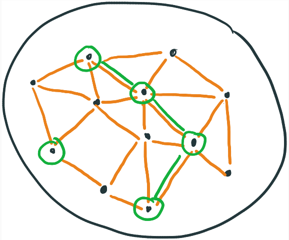
Dominating Sets to the Rescue

Goal:

- Maintain k-bounded MIS in neighborhood graph G_R with total recourse poly(k)
- Since input graph is incremental, neighborhood graph is incremental with $O(n^2)$ insertions
- Known dynamic MIS algorithms only provide $O(1)$ (expected) recourse *per update* [Behnezhad et al. '19] [Chechik, Zhang '19]

Idea:

- Maintain dominating set S on G_R with total recourse $\tilde{O}(k)$
- Maintain k-bounded MIS C on $G_R[S]$



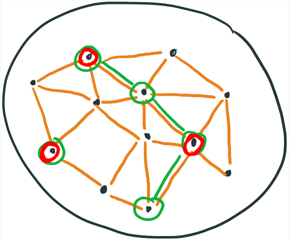
Dominating Sets to the Rescue

Goal:

- Maintain k-bounded MIS in neighborhood graph G_R with total recourse poly(k)
- Since input graph is incremental, neighborhood graph is incremental with $O(n^2)$ insertions
- Known dynamic MIS algorithms only provide $O(1)$ (expected) recourse *per update* [Behnezhad et al. '19] [Chechik, Zhang '19]

Idea:

- Maintain dominating set S on G_R with total recourse $\tilde{O}(k)$
- Maintain k-bounded MIS C on $G_R[S]$



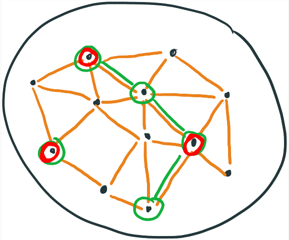
Dominating Sets to the Rescue

Goal:

- Maintain k-bounded MIS in neighborhood graph G_R with total recourse poly(k)
- Since input graph is incremental, neighborhood graph is incremental with $O(n^2)$ insertions
- Known dynamic MIS algorithms only provide $O(1)$ (expected) recourse *per update* [Behnezhad et al. '19] [Chechik, Zhang '19]

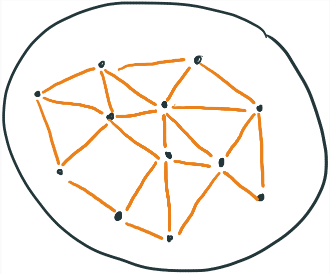
Idea:

- Maintain dominating set S on G_R with total recourse $\tilde{O}(k)$
- Maintain k-bounded MIS C on $G_R[S]$
- Every node at distance ≤ 2 to a center in G_R, and thus at distance $\leq 2R$ in G $\rightarrow (4+\epsilon)$-approximation



Dominating Sets via Random Sampling

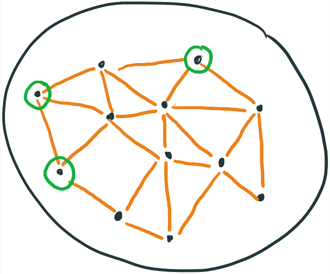
Static sampling algorithm:



Dominating Sets via Random Sampling

Static sampling algorithm:

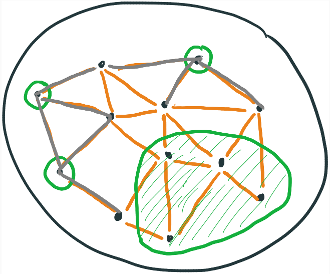
- Sample $\Theta(k \log n)$ nodes uniformly at random, add them to S



Dominating Sets via Random Sampling

Static sampling algorithm:

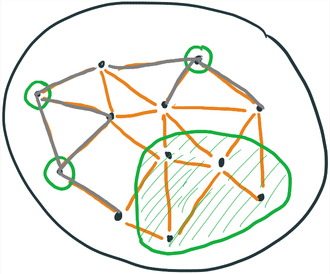
- Sample $\Theta(k \log n)$ nodes uniformly at random, add them to S
- Remove sampled nodes and neighbors from graph



Dominating Sets via Random Sampling

Static sampling algorithm:

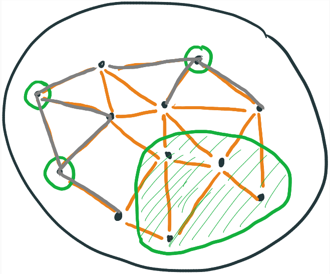
- Sample $\Theta(k \log n)$ nodes uniformly at random, add them to S
- Remove sampled nodes and neighbors from graph
- If more than $\frac{n}{2}$ nodes left: return "no MIS of size $\leq k$"



Dominating Sets via Random Sampling

Static sampling algorithm:

- Sample $\Theta(k \log n)$ nodes uniformly at random, add them to S
- Remove sampled nodes and neighbors from graph
- If more than $\frac{n}{2}$ nodes left: return "no MIS of size $\leq k$"
- Iterate until $\Theta(k \log n)$ nodes left (and add them to S)



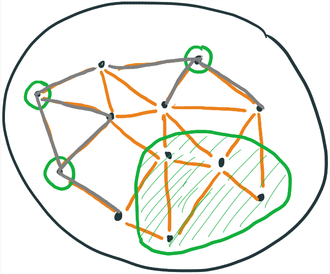
Dominating Sets via Random Sampling

Static sampling algorithm:

- Sample $\Theta(k \log n)$ nodes uniformly at random, add them to S
- Remove sampled nodes and neighbors from graph
- If more than $\frac{n}{2}$ nodes left: return "no MIS of size $\leq k$"
- Iterate until $\Theta(k \log n)$ nodes left (and add them to S)

→ Algorithm only needs access to edges incident on S in G_R

→ Incremental algorithm very similar



Bounding Size of Dominating Set

Lemma

With high probability: If graph G'_R at the end of an iteration has more than $n/2$ nodes, then G_R has no MIS of size $\leq k$.

Bounding Size of Dominating Set

Lemma

With high probability: If graph G'_R at the end of an iteration has more than $n/2$ nodes, then G_R has no MIS of size $\leq k$.

Proof by contradiction:

- Suppose G_R has MIS C of size $\leq k$

Bounding Size of Dominating Set

Lemma

With high probability: If graph G_R' at the end of an iteration has more than $n/2$ nodes, then G_R has no MIS of size $\leq k$.

Proof by contradiction:

- Suppose G_R has MIS C of size $\leq k$
- $C \cap V(G_R')$ is also an MIS in G_R'

Bounding Size of Dominating Set

Lemma

With high probability: If graph G'_R at the end of an iteration has more than $n/2$ nodes, then G_R has no MIS of size $\leq k$.

Proof by contradiction:

- Suppose G_R has MIS C of size $\leq k$
- $C \cap V(G'_R)$ is also an MIS in G'_R
- Number of nodes in G'_R is at most $|C| \times$ maximum degree in G'_R

Bounding Size of Dominating Set

Lemma

With high probability: If graph G'_R at the end of an iteration has more than $n/2$ nodes, then G_R has no MIS of size $\leq k$.

Proof by contradiction:

- Suppose G_R has MIS C of size $\leq k$
- $C \cap V(G'_R)$ is also an MIS in G'_R
- Number of nodes in G'_R is at most $|C| \times$ maximum degree in G'_R
- By random sampling ("hitting set"): maximum degree $\leq \frac{n}{2k}$

Bounding Size of Dominating Set

Lemma

With high probability: If graph G'_R at the end of an iteration has more than $n/2$ nodes, then G_R has no MIS of size $\leq k$.

Proof by contradiction:

- Suppose G_R has MIS C of size $\leq k$
- $C \cap V(G'_R)$ is also an MIS in G'_R
- Number of nodes in G'_R is at most $|C| \times$ maximum degree in G'_R
- By random sampling (“hitting set”): maximum degree $\leq \frac{n}{2k}$
- Thus, G'_R has at most $k \times \frac{n}{2k} = \frac{n}{2}$ nodes. Contradiction

Bounding Size of Dominating Set

Lemma

With high probability: If graph G_R' at the end of an iteration has more than $n/2$ nodes, then G_R has no MIS of size $\leq k$.

Proof by contradiction:

- Suppose G_R has MIS C of size $\leq k$
- $C \cap V(G_R')$ is also an MIS in G_R'
- Number of nodes in G_R' is at most $|C| \times$ maximum degree in G_R'
- By random sampling ("hitting set"): maximum degree $\leq \frac{n}{2k}$
- Thus, G_R' has at most $k \times \frac{n}{2k} = \frac{n}{2}$ nodes. Contradiction

Consequence:

- $O(\log n)$ iterations of sampling procedure
- S has $\tilde{O}(k)$ nodes

An Open Problem

Question

Can we efficiently maintain a k-bounded MIS with total recourse poly(k)?

An Open Problem

Question

Can we efficiently maintain a k-bounded MIS with total recourse $\text{poly}(k)$?

Random edge insertions analogy:

- Each endpoint is in k-bounded MIS with probability $\frac{k}{n}$

An Open Problem

Question

Can we efficiently maintain a k-bounded MIS with total recourse poly(k)?

Random edge insertions analogy:

- Each endpoint is in k-bounded MIS with probability $\frac{k}{n}$
  $\rightarrow$ at most n^2 insertions with “conflict” probability $\frac{k^2}{n^2}$ each

An Open Problem

Question

Can we efficiently maintain a k-bounded MIS with total recourse poly(k)?

Random edge insertions analogy:

- Each endpoint is in k-bounded MIS with probability $\frac{k}{n}$
  $\rightarrow$ at most n^2 insertions with "conflict" probability $\frac{k^2}{n^2}$ each
- Suggests that a total recourse of $\tilde{O}(k^2)$ might be achievable against an oblivious adversary

An Open Problem

Question

Can we efficiently maintain a k-bounded MIS with total recourse poly(k)?

Random edge insertions analogy:

- Each endpoint is in k-bounded MIS with probability $\frac{k}{n}$
  $\rightarrow$ at most n^2 insertions with "conflict" probability $\frac{k^2}{n^2}$ each
- Suggests that a total recourse of $\tilde{O}(k^2)$ might be achievable against an oblivious adversary
- Could potentially carry over to $(2+\epsilon)$-approximate incremental k-center with amortized update time $\tilde{O}(k^2)$

An Open Problem

Question

Can we efficiently maintain a k-bounded MIS with total recourse poly(k)?

Random edge insertions analogy:

- Each endpoint is in k-bounded MIS with probability $\frac{k}{n}$
  $\rightarrow$ at most n^2 insertions with "conflict" probability $\frac{k^2}{n^2}$ each
- Suggests that a total recourse of $\tilde{O}(k^2)$ might be achievable against an oblivious adversary
- Could potentially carry over to $(2+\epsilon)$-approximate incremental k-center with amortized update time $\tilde{O}(k^2)$
- Attention: Recourse guarantee is needed for dense neighborhood graph

Conclusion

Remarks

Nice application of dynamic approximate SSSP

Path-reporting seems to be less relevant in this context

Incremental was the difficult question for this problem

Incremental model more relevant than we usually consider

Interesting question about dynamic MIS suddenly shows up

Future Work

Consider other clustering objectives in graphs

Engineer dynamic approximate SSSP algorithm

Future Work

Consider other clustering objectives in graphs

Engineer dynamic approximate SSSP algorithm

Thank you!