An Update to Dynamic Graph Algorithms

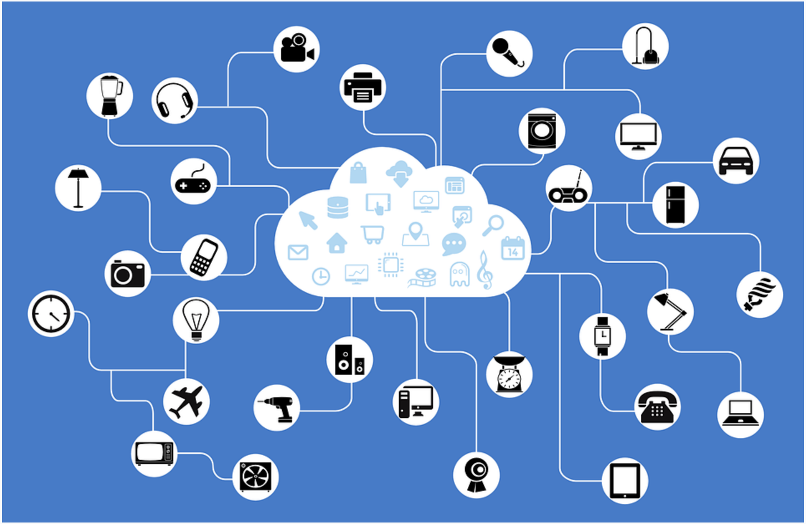
Sebastian Forster, né Krinninger

Paris Lodron University Salzburg

@ACSD 2024

Supported by the EXDIGIT (Excellence in Digital Sciences and Interdisciplinary Technology) project, funded by Land Salzburg under grant number 20204-WISS/263/6-6022, and 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).

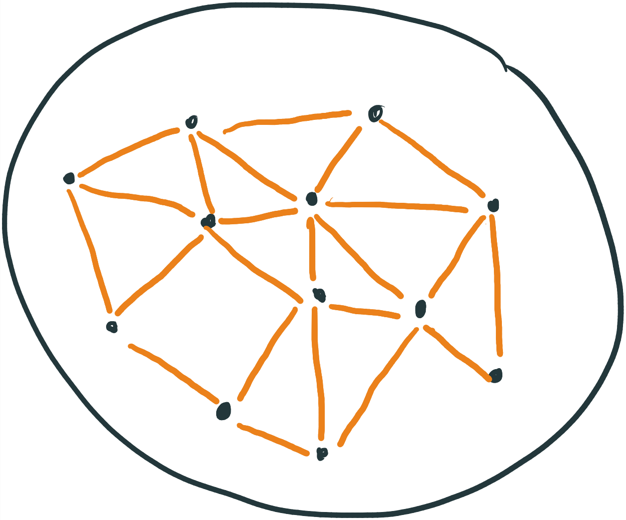
Graphs are Everywhere



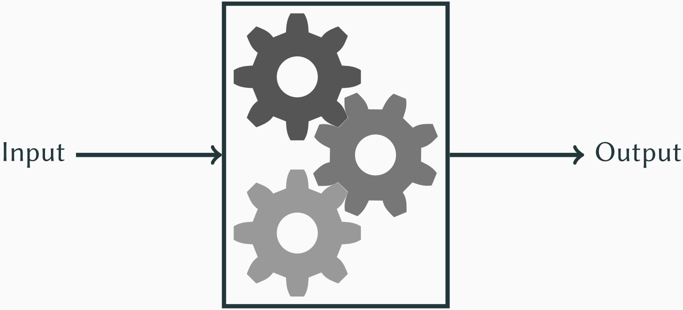
Graphs are Everywhere

Graphs are Everywhere

Graphs are Everywhere



Static Algorithms



Dynamic Environments

Dynamic Environments

≈ 50 % of applications for big graphs are dynamic [Sahu et al. ’17]

Running Time

Goal

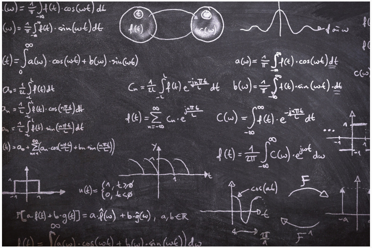
Design algorithms that react **quickly** to changes in the input data

Running Time

Goal

Design algorithms that react **quickly** to changes in the input data

Measurement



Mathematical analysis

Warm-Up: Moving Average

Time series: $s_1, s_2, \dots, s_n$

Mean of last k values:

$\bar{s}_{n,k} = \frac{s_n + s_{n-1} + \dots + s_{n-k+1}}{k}$

$\rightarrow$ k arithmetic operations

Warm-Up: Moving Average

Time series: $s_1, s_2, \dots, s_n$

Mean of last k values:

$\bar{s}_{n,k} = \frac{s_n + s_{n-1} + \cdots + s_{n-k+1}}{k}$

$\rightarrow k$ arithmetic operations

Equivalent formula:

$\bar{s}_{n,k} = \bar{s}_{n-1,k} + \frac{1}{k}(s_n - s_{n-k})$

Warm-Up: Moving Average

Time series: $s_1, s_2, \ldots, s_n$

Mean of last k values:

$\bar{s}_{n,k} = \frac{s_n + s_{n-1} + \cdots + s_{n-k+1}}{k}$

$\rightarrow$ k arithmetic operations

Equivalent formula:

$\bar{s}_{n,k} = \bar{s}_{n-1,k} + \frac{1}{k}(s_n - s_{n-k})$

$\rightarrow$ 3 arithmetic operations

Efficiency gain!

Status Quo

Success Story:

- Fast dynamic graph algorithms for fundamental, “textbook” problems: connectivity, shortest paths, matching, ...
- Sophisticated mathematical tools and techniques
- Dynamic graph algorithms facilitate breakthroughs in combinatorial optimization [Chen et al. ’22]

Status Quo

Success Story:

- Fast dynamic graph algorithms for fundamental, "textbook" problems: connectivity, shortest paths, matching, ...
- Sophisticated mathematical tools and techniques
- Dynamic graph algorithms facilitate breakthroughs in combinatorial optimization [Chen et al. '22]

Problem: (Too) little real-world impact

Status Quo

Success Story:

- Fast dynamic graph algorithms for fundamental, “textbook” problems: connectivity, shortest paths, matching, ...
- Sophisticated mathematical tools and techniques
- Dynamic graph algorithms facilitate breakthroughs in combinatorial optimization [Chen et al. ’22]

Problem: (Too) little real-world impact

- Complicated algorithms
- Lack of (scalable) implementations
- Practitioners interested in wider array of problems

Towards Dynamic Graph Mining Algorithms

Vision

Systematically transfer technology developed for dynamic graph algorithms to graph mining and learning domain

Towards Dynamic Graph Mining Algorithms

Vision

Systematically transfer technology developed for dynamic graph algorithms to graph mining and learning domain

Focus on Relevant Problems:

- Centrality
- Clustering
- Pattern (subgraph) detection
- ...

Recent survey [Hanauer, Henzinger, Schulz '22] reveals blind spots

Towards Dynamic Graph Mining Algorithms

Vision

Systematically transfer technology developed for dynamic graph algorithms to graph mining and learning domain

Focus on Relevant Problems:

- Centrality
- Clustering
- Pattern (subgraph) detection
- ...

Recent survey [Hanauer, Henzinger, Schulz '22] reveals blind spots

Towards Dynamic Graph Mining Algorithms

Vision

Systematically transfer technology developed for dynamic graph algorithms to graph mining and learning domain

Focus on Relevant Problems:

- Centrality
- Clustering
- Pattern (subgraph) detection
- ...

Recent survey [Hanauer, Henzinger, Schulz '22] reveals blind spots

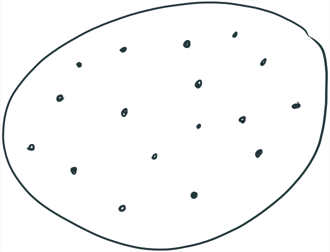
Integrated Pipeline

Algorithm design → Algorithm engineering → Applications

Case Study: k-Center Clustering

k-Center Problem

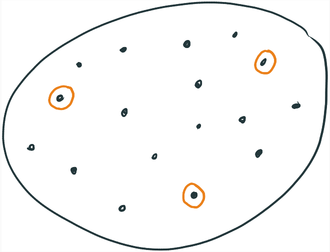
Given a metric space, select k points as set of centers C such that the maximum distance $d(C, v)$ of any node v to its closest center is minimized.



Case Study: 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.

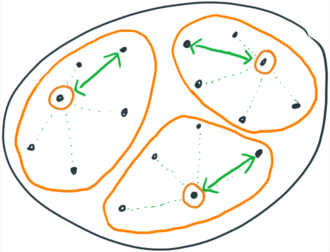


Case Study: k-Center Clustering

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

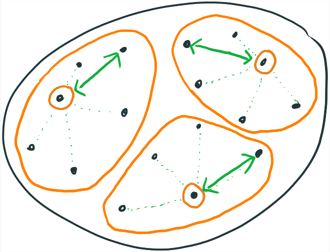


Case Study: k-Center Clustering

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
- Problem is NP-hard to approximate within a factor of $2 - \epsilon$

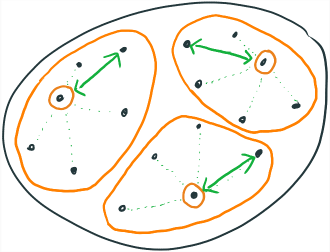


Case Study: k-Center Clustering

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
- Problem is NP-hard to approximate within a factor of $2 - \epsilon$
- Prior work for dynamic point sets [Chan, Gourqin, Sozio '18] [Bateni et al. '23]



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

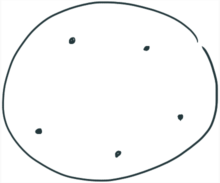
Question

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:

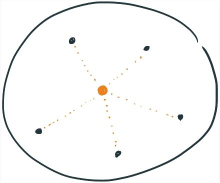
- Point insertions and deletions



Dynamic Model

Dynamic Point Sets:

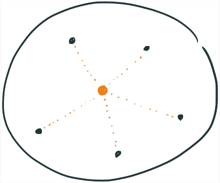
- Point insertions and deletions



Dynamic Model

Dynamic Point Sets:

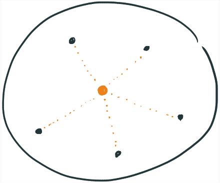
- Point insertions and deletions
- Query access to metric



Dynamic Model

Dynamic Point Sets:

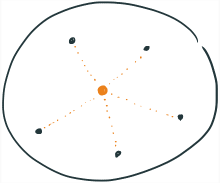
- Point insertions and deletions
- Query access to metric
- Metric extends/reduces



Dynamic Model

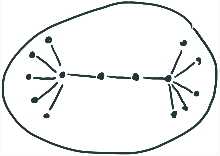
Dynamic Point Sets:

- Point insertions and deletions
- Query access to metric
- Metric extends/reduces



Dynamic Graphs:

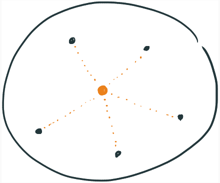
- Edge insertions and deletions



Dynamic Model

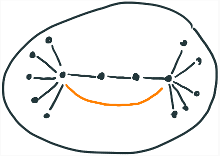
Dynamic Point Sets:

- Point insertions and deletions
- Query access to metric
- Metric extends/reduces



Dynamic Graphs:

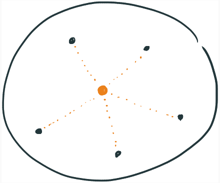
- Edge insertions and deletions



Dynamic Model

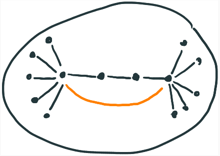
Dynamic Point Sets:

- Point insertions and deletions
- Query access to metric
- Metric extends/reduces



Dynamic Graphs:

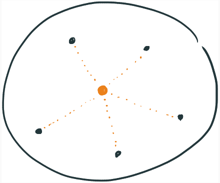
- Edge insertions and deletions
- Distances not given for free



Dynamic Model

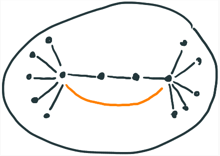
Dynamic Point Sets:

- Point insertions and deletions
- Query access to metric
- Metric extends/reduces



Dynamic Graphs:

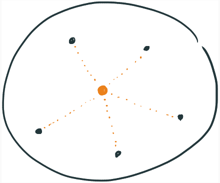
- Edge insertions and deletions
- Distances not given for free
- Metric shrinks/expands



Dynamic Model

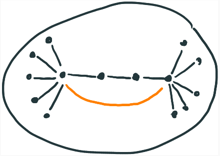
Dynamic Point Sets:

- Point insertions and deletions
- Query access to metric
- Metric extends/reduces



Dynamic Graphs:

- Edge insertions and deletions
- Distances not given for free
- Metric shrinks/expands



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] [**F** and Skarlatos ’24]

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] [**F** and Skarlatos '24]

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

Our Results I: Fully Dynamic

Theorem (Cruciani, F, Goranci, Nazari, Skarlatos '24)

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 ’24)

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 '24)

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

Our Results II: Partially Dynamic

Theorem (Cruciani, F, Goranci, Nazari, Skarlatos '24)

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

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

[Henzinger, **K**, Nanongkai '14] [Bernstein, Probst G., Saranurak '21]

Our Results II: Partially Dynamic

Theorem (Cruciani, F, Goranci, Nazari, Skarlatos '24)

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

Update time for decremental $(1+\epsilon)$-approximate SSSP: $n^{o(1)}$
[Henzinger, **K**, Nanongkai '14] [Bernstein, Probst G., Saranurak '21]

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

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

Our Results II: Partially Dynamic

Theorem (Cruciani, F, Goranci, Nazari, Skarlatos ’24)

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

Update time for decremental $(1+\epsilon)$-approximate SSSP: $n^{o(1)}$
[Henzinger, **K**, Nanongkai ’14] [Bernstein, Probst G., Saranurak ’21]

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

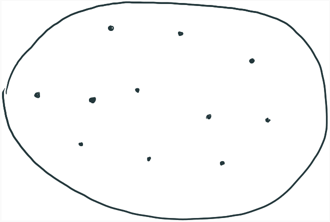
There is a randomized incremental (= insertions-only) $(4+\epsilon)$-approximate k-center algorithm with amortized update time $kn^{o(1)}$ that is correct against an oblivious adversary for any constant ϵ.

Update time for incremental $(1+\epsilon)$-approximate SSSP: $n^{o(1)}$
[implicit in Henzinger, **K**, Nanongkai ’14]

Reminder: Gonzalez's Algorithm

Gonzalez's Algorithm [Gonzalez '85]

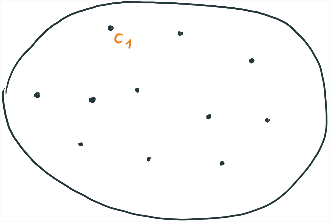
1. Initialize $C = \{v\}$ with arbitrary first center
2. While $|C| < k$, add node v maximizing $d(C, v)$ to C



Reminder: Gonzalez's Algorithm

Gonzalez's Algorithm [Gonzalez '85]

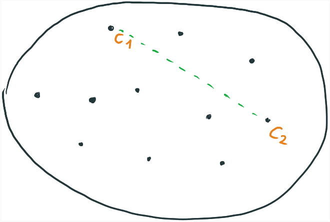
1. Initialize $C = \{v\}$ with arbitrary first center
2. While $|C| < k$, add node v maximizing $d(C, v)$ to C



Reminder: Gonzalez's Algorithm

Gonzalez's Algorithm [Gonzalez '85]

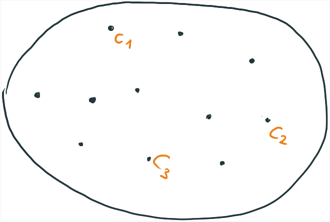
1. Initialize $C = \{v\}$ with arbitrary first center
2. While $|C| < k$, add node v maximizing $d(C, v)$ to C



Reminder: Gonzalez's Algorithm

Gonzalez's Algorithm [Gonzalez '85]

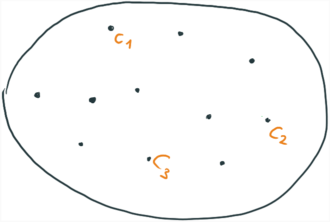
1. Initialize $C = \{v\}$ with arbitrary first center
2. While $|C| < k$, add node v maximizing $d(C, v)$ to C



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

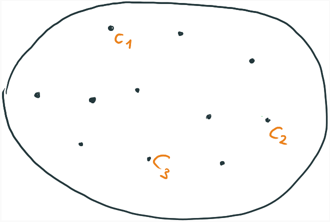


This gives a 2-approximation

Reminder: Gonzalez's Algorithm

Gonzalez's Algorithm [Gonzalez '85]

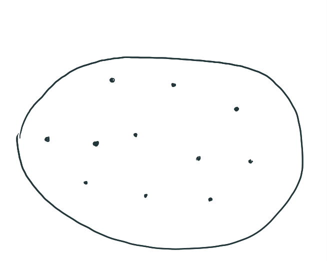
1. Initialize $C = \{v\}$ with arbitrary first center
2. While $|C| < k$, add node v maximizing $d(C, v)$ to C



This gives a 2-approximation

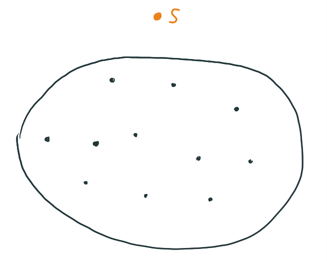
If $d(C, v)$ is within factor $1 + \epsilon$ of maximum, this gives $(2 + \epsilon)$-approximation

Fully Dynamic Algorithm: Simulating Gonzalez's Algorithm



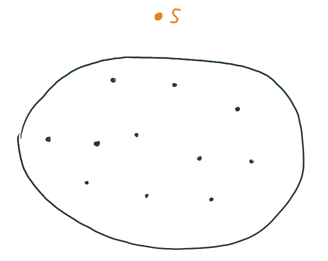
Fully Dynamic Algorithm: Simulating Gonzalez's Algorithm

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



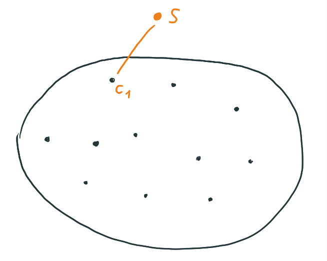
Fully Dynamic Algorithm: Simulating Gonzalez's Algorithm

- 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 distance data structure



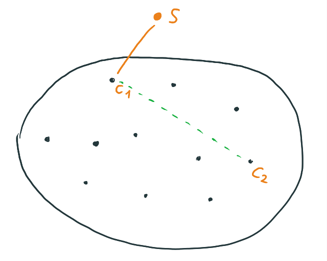
Fully Dynamic Algorithm: Simulating Gonzalez's Algorithm

- 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 distance data structure
    - Initialize $C = \{v\}$ with arbitrary first center and connect it to s



Fully Dynamic Algorithm: Simulating Gonzalez's Algorithm

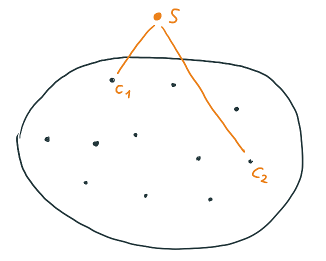
- Add artificial "super-source" s
- Maintain $(1+\epsilon)$-approximate single-source distances from s with a fully dynamic algorithm



- After every update to graph:
  - Forward update to distance 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

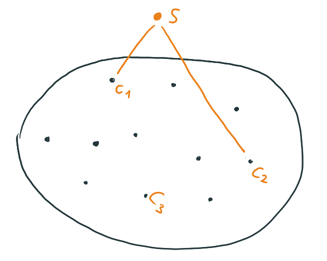
Fully Dynamic Algorithm: Simulating Gonzalez's Algorithm

- 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 distance 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



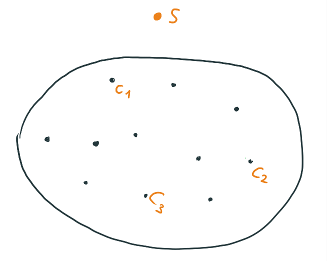
Fully Dynamic Algorithm: Simulating Gonzalez's Algorithm

- 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 distance 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



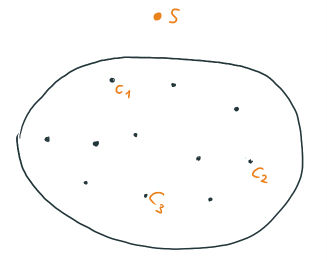
Fully Dynamic Algorithm: Simulating Gonzalez's Algorithm

- 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 distance 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



Fully Dynamic Algorithm: 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 distance 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: $O(k \cdot U_{\text{SSSP}}(n))$

Outlook: Towards Dynamic Graph Mining Algorithms

Challenges:

- Experimental methodology not fully established
- Widespread use of heuristics in mining and learning domain
- Finding non-industrial applications

Outlook: Towards Dynamic Graph Mining Algorithms

Challenges:

- Experimental methodology not fully established
- Widespread use of heuristics in mining and learning domain
- Finding non-industrial applications

Opportunities:

- Real-time data analysis
- Interesting research problems

Thanks for your attention!

forster@cs.sbg.ac.at

https://bda.cs.plus.ac.at