Stable Approximation Schemes

Mark de Berg (TU Eindhoven)

TU/e

NET WORKS

Talk Overview

Stable Approximation Schemes

Basic Terminology and Relations to Existing Concepts

TU/e

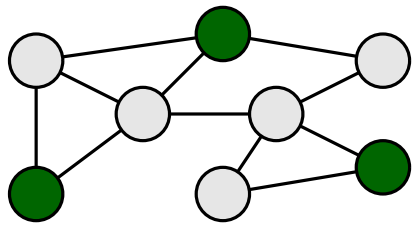
NET WORKS

Stable Algorithms

Optimization problems

- compute optimal solution
  (minimum cost, or maximum profit)
- many optimization problems are NP-hard

  $\Longrightarrow$ focus on approximation algorithms
  or parameterized algorithms
  or subexponential algorithms



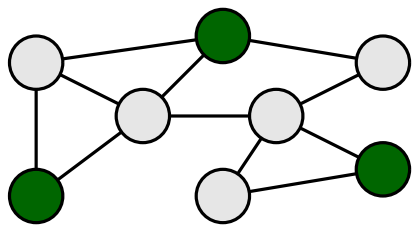
Independent Set

Stable Algorithms

Optimization problems

- compute optimal solution
  (minimum cost, or maximum profit)
- many optimization problems are NP-hard

  $\Longrightarrow$ focus on approximation algorithms
  or parameterized algorithms
  or subexponential algorithms



Independent Set

What if

- cost is in "executing" the solution rather than computing it
- input is changing over time

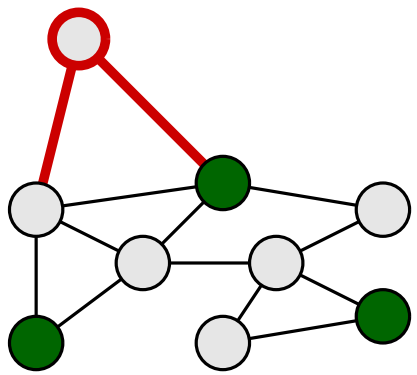
We want to maintain a stable solution, which does not change much when input changes but is still close to optimal

Stable Algorithms

Optimization problems

- compute optimal solution
  (minimum cost, or maximum profit)
- many optimization problems are NP-hard

  $\Longrightarrow$ focus on approximation algorithms
  or parameterized algorithms
  or subexponential algorithms



Independent Set

What if

- cost is in "executing" the solution rather than computing it
- input is changing over time

We want to maintain a stable solution, which does not change much when input changes but is still close to optimal

Stable Algorithms

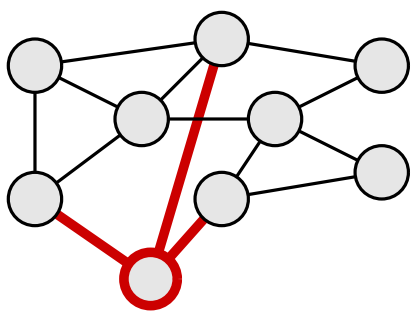
The framework

- define dynamic model
  - insertion-only, or fully dynamic, or deletion-only
  - define what an insertion/deletion means

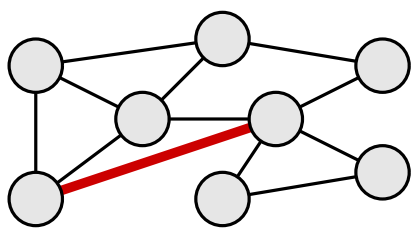
The framework

- define dynamic model
  - insertion-only, or fully dynamic, or deletion-only
  - define what an insertion/deletion means

for graphs:



vertex-arrival model



edge-arrival model

Stable Algorithms

The framework

- define dynamic model
  - insertion-only, or fully dynamic, or deletion-only
  - define what an insertion/deletion means

Stable Algorithms

The framework

- define dynamic model
  - insertion-only, or fully dynamic, or deletion-only
  - define what an insertion/deletion means

- define "edit distance" between any two (not necessarily optimal) solutions

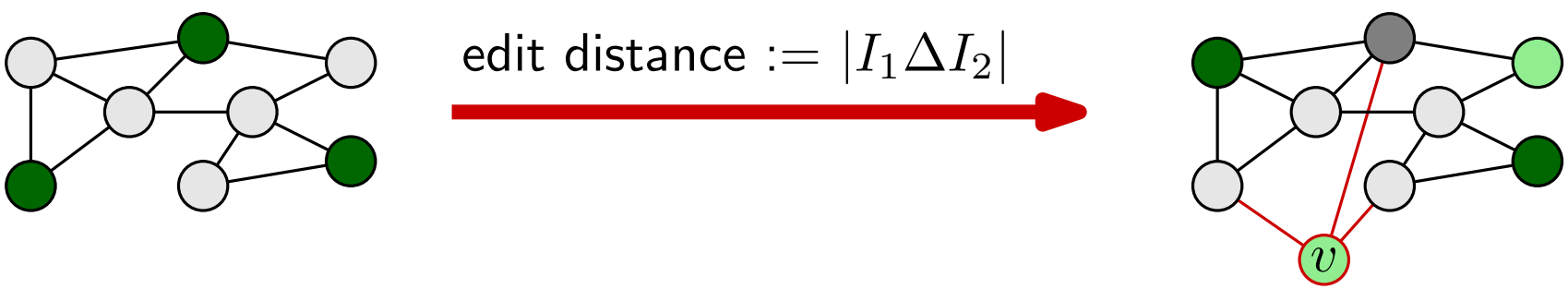
Stable Algorithms

The framework

- define dynamic model
  - insertion-only, or fully dynamic, or deletion-only
  - define what an insertion/deletion means

- define "edit distance" between any two (not necessarily optimal) solutions

Example: Independent Set



solution I_1 before
arrival of vertex v

solution I_2 after
arrival of vertex v

Stable Algorithms

The framework

- define dynamic model
  - insertion-only, or fully dynamic, or deletion-only
  - define what an insertion/deletion means

- define "edit distance" between any two (not necessarily optimal) solutions

Stable Algorithms

The framework

- define dynamic model
  - insertion-only, or fully dynamic, or deletion-only
  - define what an insertion/deletion means

- define "edit distance" between any two (not necessarily optimal) solutions

Definition. A k-stable α-approximation algorithm maintains a feasible solution such that

- the edit distance between the solutions before and after each update (insertion or deletion) is at most k
- the maintained solution is an α-approximation of an optimal solution at all times

Stable Approximation Schemes (SAS)

We would like a trade-off between stability and approximation ratio.

Definition. A stable approximation scheme (SAS) is an algorithm that, for any given parameter $\varepsilon > 0$,

- is k_ε-stable, where k_ε only depends on ε (not on the input size)
- gives a $(1+\varepsilon)$-approximation

Stable Algorithms and Stable Approximation Schemes

Related concepts

- online algorithms, which process sequence σ of insertions
  - decisions are irrevocable
  - competitive ratio instead of approximation ratio

Stable Algorithms and Stable Approximation Schemes

Related concepts

- online algorithms, which process sequence σ of insertions
  - decisions are irrevocable
  - competitive ratio instead of approximation ratio

$\textsc{Alg}(\sigma)$ = value of solution computed by online algorithm

$\textsc{Opt}_{\text{offline}}(\sigma)$ = value of solution computed by optimal offline algorithm (algorithm knows σ in advance, but must still handle insertions in online manner)

$\textsc{Opt}(\sigma)$ = value of optimal solution on final input (static optimum)

Stable Algorithms and Stable Approximation Schemes

Related concepts

- online algorithms, which process sequence σ of insertions
  - decisions are irrevocable
  - competitive ratio instead of approximation ratio

$\textsc{Alg}(\sigma)$ = value of solution computed by online algorithm

$\textsc{Opt}_{\text{offline}}(\sigma)$ = value of solution computed by optimal offline algorithm (algorithm knows σ in advance, but must still handle insertions in online manner)

$\textsc{Opt}(\sigma)$ = value of optimal solution on final input (static optimum)

$$\text{competitive ratio} \approx \frac{\textsc{Alg}(\sigma)}{\textsc{Opt}_{\text{offline}}(\sigma)}$$

$$\text{approximation ratio} = \frac{\textsc{Alg}(\sigma)}{\textsc{Opt}(\sigma)}$$

Stable Algorithms and Stable Approximation Schemes

Related concepts

- online algorithms, which process sequence σ of insertions
  - decisions are irrevocable
  - competitive ratio instead of approximation ratio

Related concepts

- online algorithms, which process sequence σ of insertions
  - decisions are irrevocable
  - competitive ratio instead of approximation ratio

- online algorithms with bounded recourse
  - recourse $\approx$ stability
  - analysis often still uses competitive ratio (?)

- robust PTAS: similar to our SAS

- local-search PTAS

Stable Algorithms and Stable Approximation Schemes

Related concepts

- online algorithms, which process sequence σ of insertions
  - decisions are irrevocable
  - competitive ratio instead of approximation ratio

- online algorithms with bounded recourse
  - recourse $\approx$ stability
  - analysis often still uses competitive ratio (?)

- robust PTAS: similar to our SAS

- local-search PTAS

We typically do not care about computation time, and
focus on trade-off between stability and approximation ratio.

Stable Approximation Schemes for

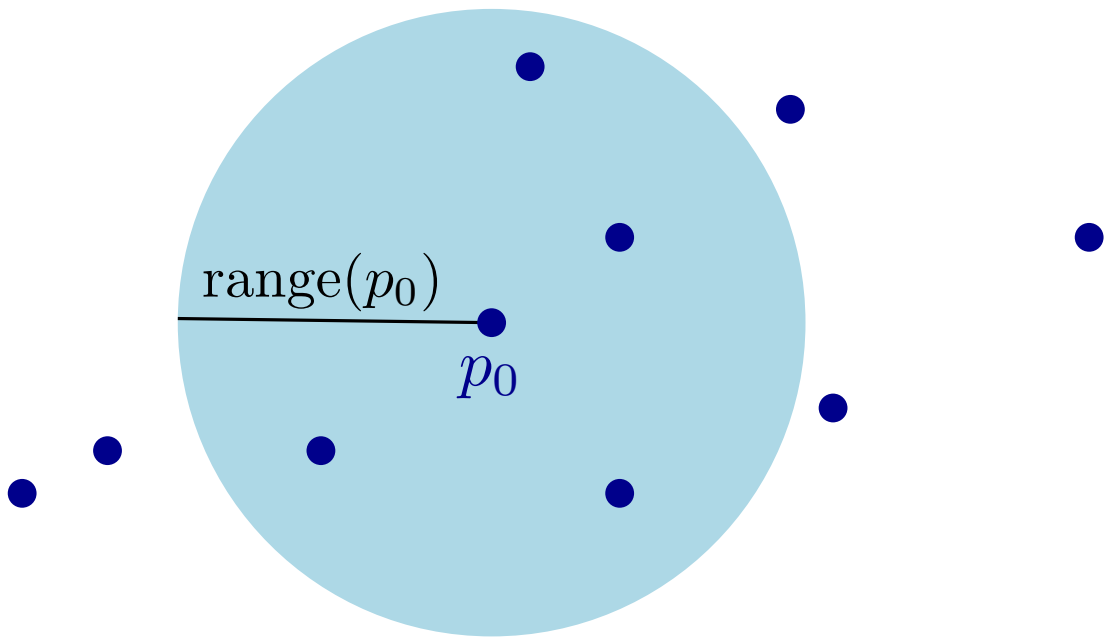
Broadcast Range Assignment

TU/e

NET WORKS

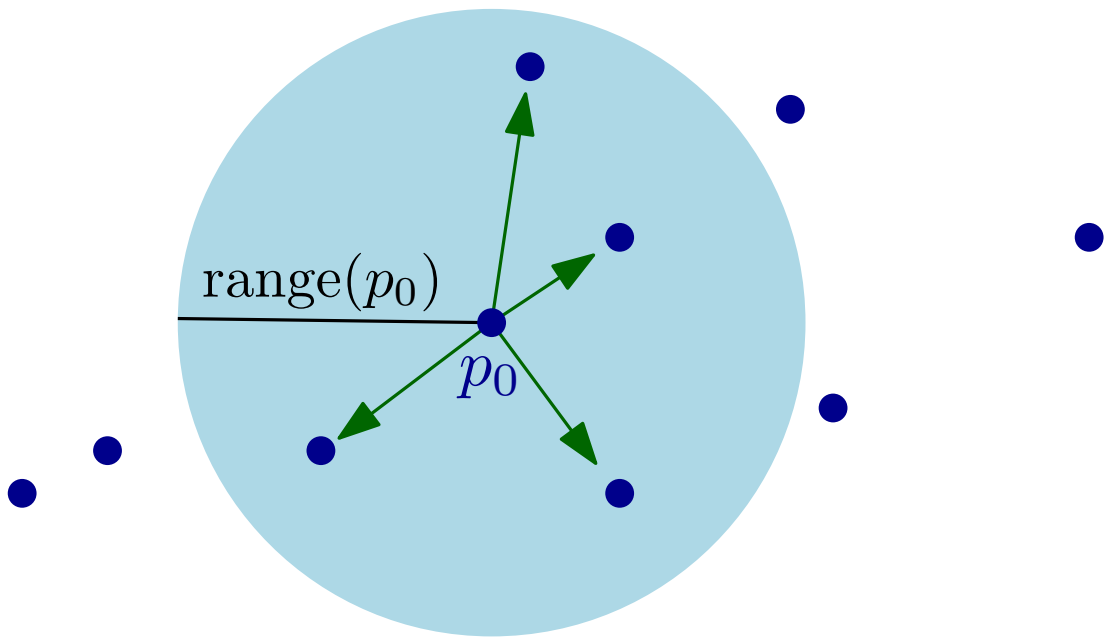
The range-assignment problem

- $P = \{p_0, \ldots, p_{n-1}\}$: set of n points in $\mathbb{R}^2$ (devices in wireless network)
- $\text{range}(p_i)$: transmission range of p_i



The range-assignment problem

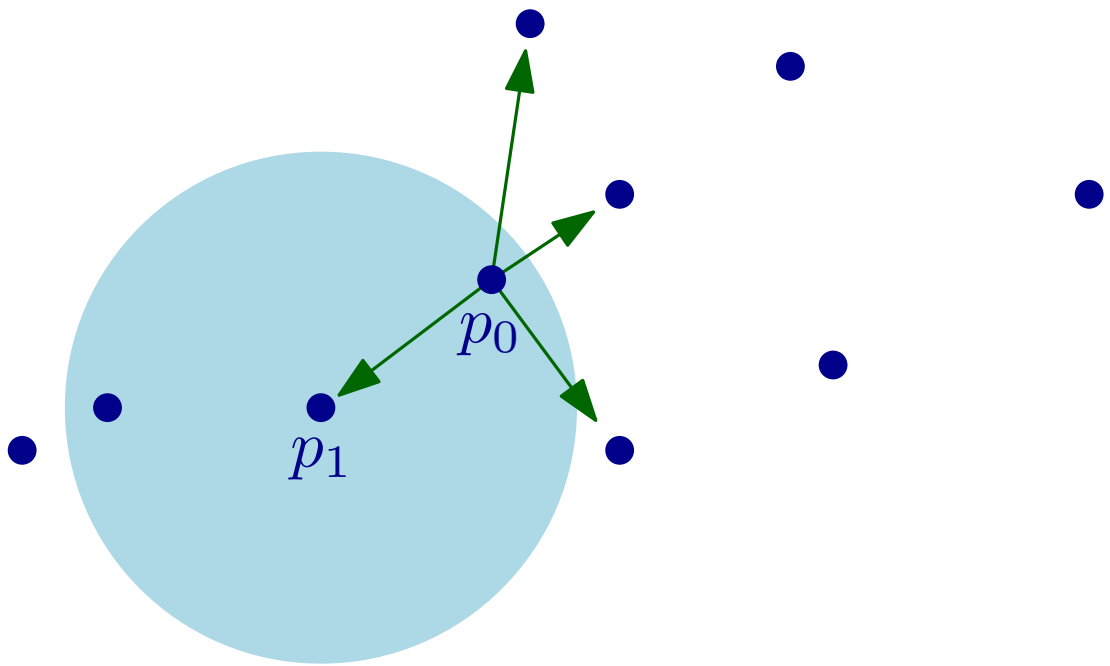
- $P = \{p_0, \ldots, p_{n-1}\}$: set of n points in $\mathbb{R}^2$ (devices in wireless network)
- $\text{range}(p_i)$: transmission range of p_i



p_i can send information to p_j iff $|p_i p_j| \leqslant \text{range}(p_i)$

The range-assignment problem

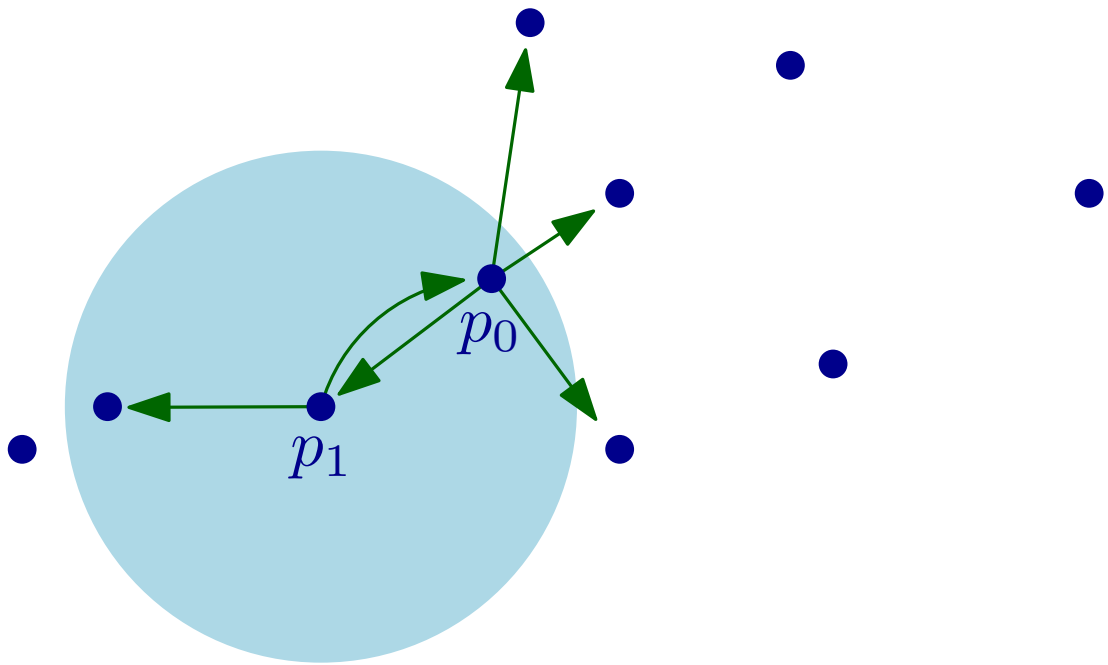
- $P = \{p_0, \ldots, p_{n-1}\}$: set of n points in $\mathbb{R}^2$ (devices in wireless network)
- $\mathrm{range}(p_i)$: transmission range of p_i



p_i can send information to p_j iff $|p_i p_j| \leqslant \mathrm{range}(p_i)$

The range-assignment problem

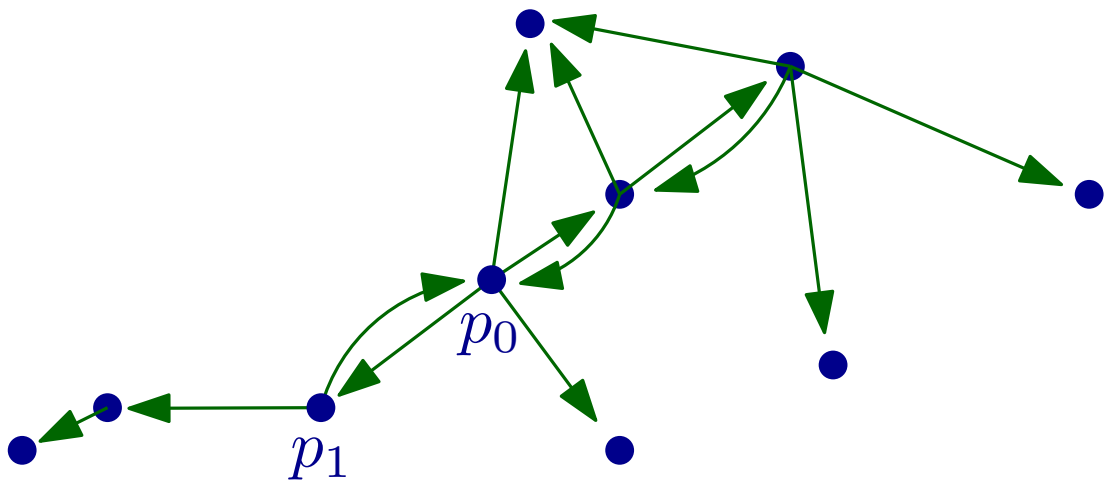
- $P = \{p_0, \ldots, p_{n-1}\}$: set of n points in $\mathbb{R}^2$ (devices in wireless network)
- $\text{range}(p_i)$: transmission range of p_i



p_i can send information to p_j iff $|p_i p_j| \leqslant \text{range}(p_i)$

The range-assignment problem

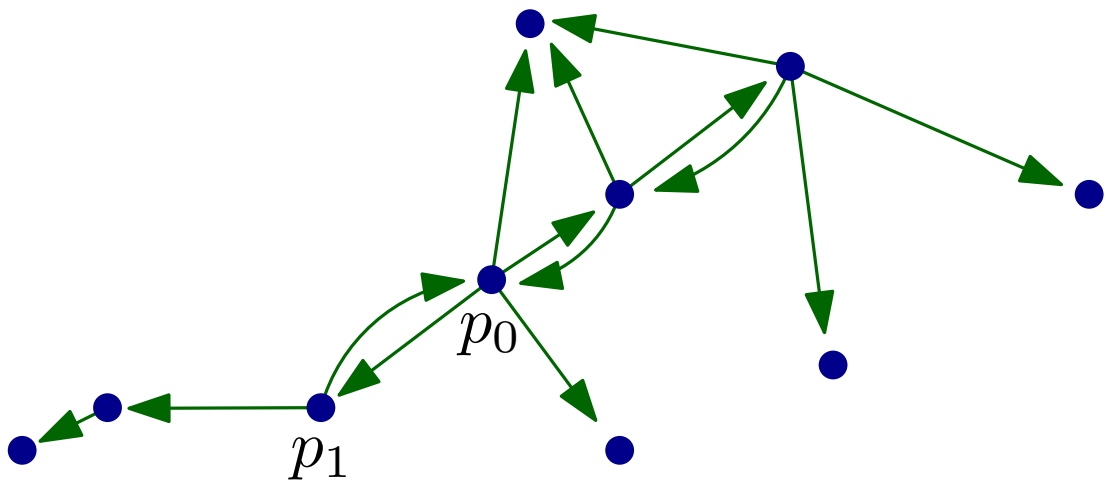
- $P = \{p_0, \ldots, p_{n-1}\}$: set of n points in $\mathbb{R}^2$ (devices in wireless network)
- $\text{range}(p_i)$: transmission range of p_i



- range assignment induces communication graph

The range-assignment problem

- $P = \{p_0, \ldots, p_{n-1}\}$: set of n points in $\mathbb{R}^2$ (devices in wireless network)
- $\text{range}(p_i)$: transmission range of p_i

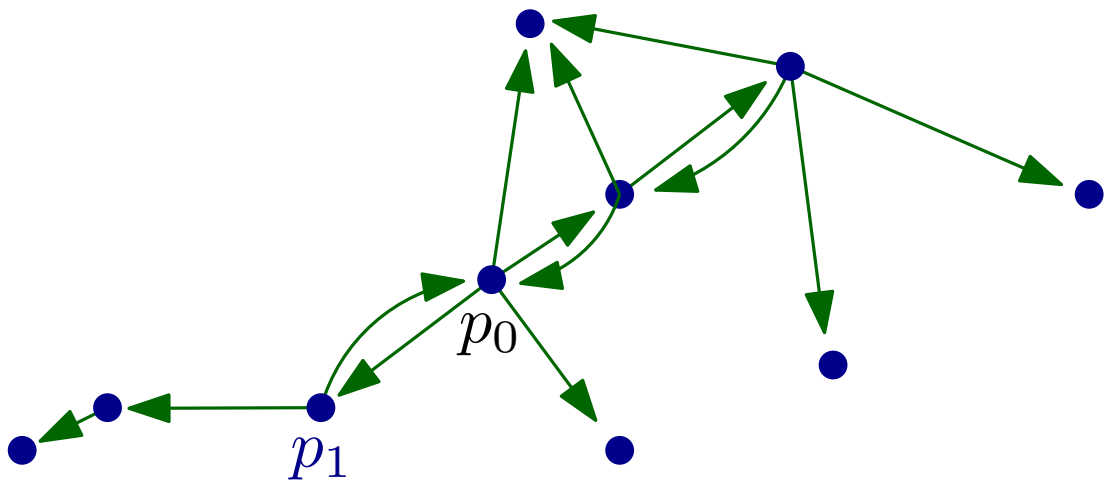


- range assignment induces communication graph
- cost of range assignment $= \sum_i \text{range}(p_i)^\alpha$

distance-power gradient
practice: $1 \leqslant \alpha \leqslant 6$
from now on: $\alpha = 2$

The range-assignment problem

- $P = \{p_0, \ldots, p_{n-1}\}$: set of n points in $\mathbb{R}^2$ (devices in wireless network)
- $\text{range}(p_i)$: transmission range of p_i



- range assignment induces communication graph
- cost of range assignment $= \sum_i \text{range}(p_i)^\alpha$

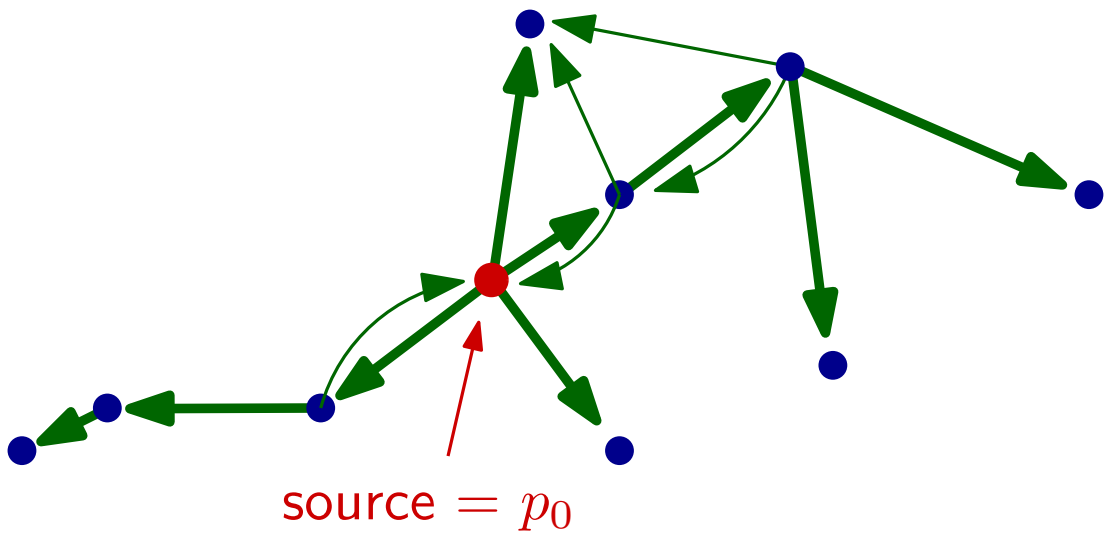
range-assignment problem: assign ranges so that communication graph has certain properties, while minimizing cost

The broadcast range-assignment problem

broadcast range-assignment problem: assign ranges so that communication graph has broadcast tree from given source p_0, while minimizing cost

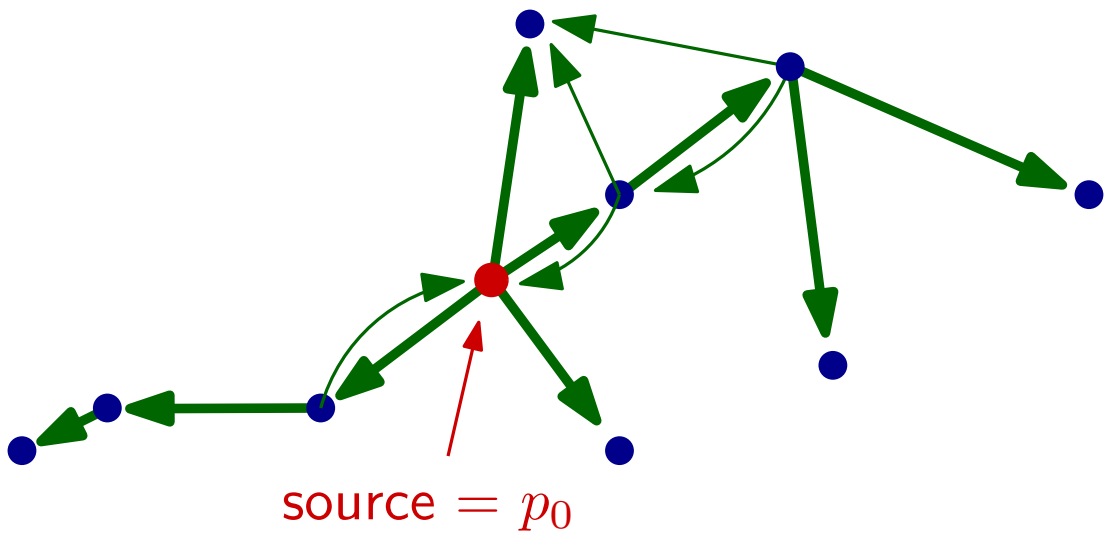
The broadcast range-assignment problem

broadcast range-assignment problem: assign ranges so that communication graph has broadcast tree from given source p_0, while minimizing cost



The broadcast range-assignment problem

broadcast range-assignment problem: assign ranges so that communication graph has broadcast tree from given source p_0, while minimizing cost



- polynomial-time solvable in $\mathbb{R}^1$ [Clementi *et al.*]
- NP-hard in $\mathbb{R}^2$, $O(1)$-approximation [Clementi *et al.*, Fuchs, Wan *et al.*]
- APX-hard in $\mathbb{R}^3$ [Fuchs]

The dynamic broadcast range-assignment problem

Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

The dynamic broadcast range-assignment problem

Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

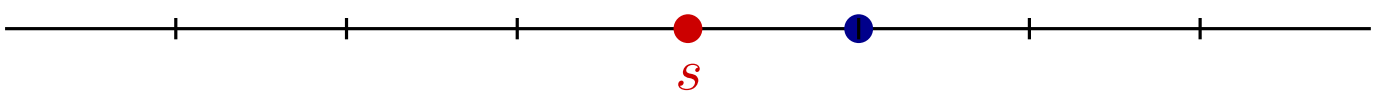
Example in $\mathbb{R}^1$ (online algorithm that does not decrease any ranges)

The dynamic broadcast range-assignment problem

Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

Example in $\mathbb{R}^1$ (online algorithm that does not decrease any ranges)

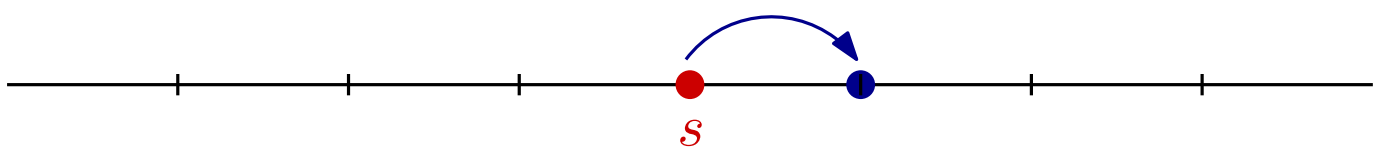


The dynamic broadcast range-assignment problem

Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

Example in $\mathbb{R}^1$ (online algorithm that does not decrease any ranges)



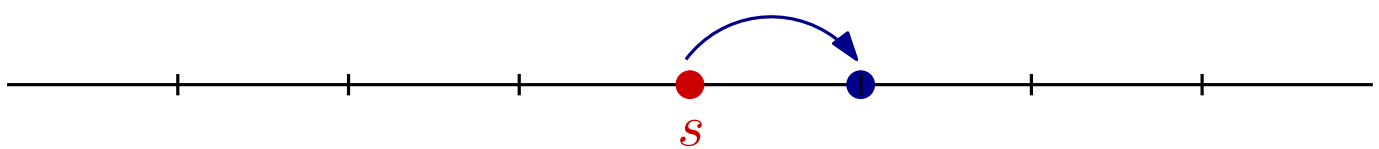
- range$(s) := 1$
- cost $= 1^2 = 1$

The dynamic broadcast range-assignment problem

Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

Example in $\mathbb{R}^1$ (online algorithm that does not decrease any ranges)

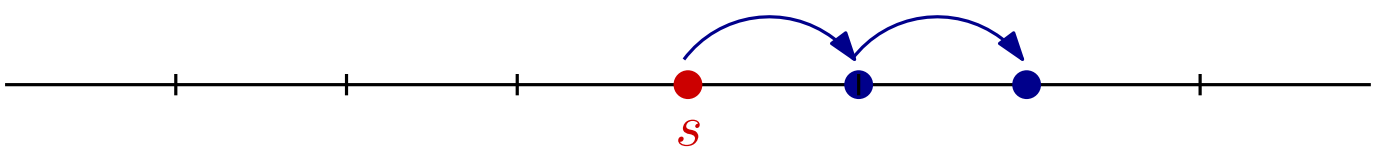


The dynamic broadcast range-assignment problem

Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

Example in $\mathbb{R}^1$ (online algorithm that does not decrease any ranges)

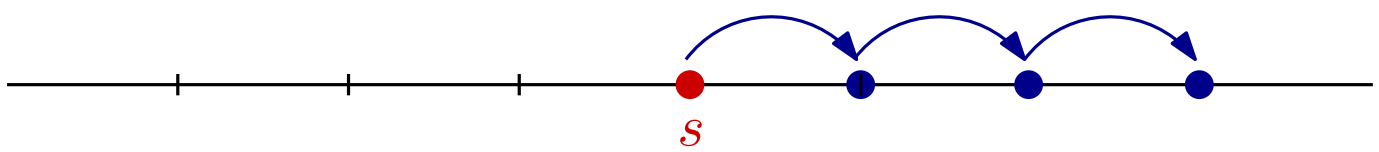


The dynamic broadcast range-assignment problem

Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

Example in $\mathbb{R}^1$ (online algorithm that does not decrease any ranges)



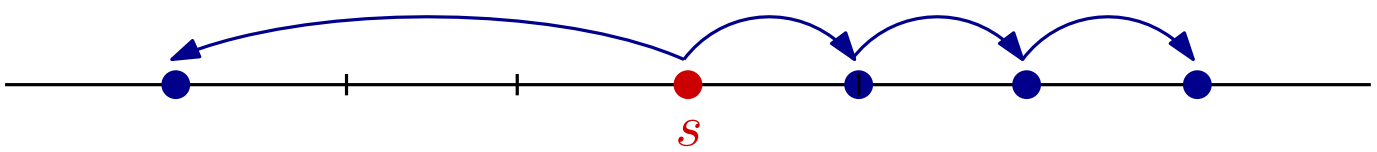
$$\text{cost} = 1^2 + 1^2 + 1^2 = 3$$

The dynamic broadcast range-assignment problem

Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

Example in $\mathbb{R}^1$ (online algorithm that does not decrease any ranges)

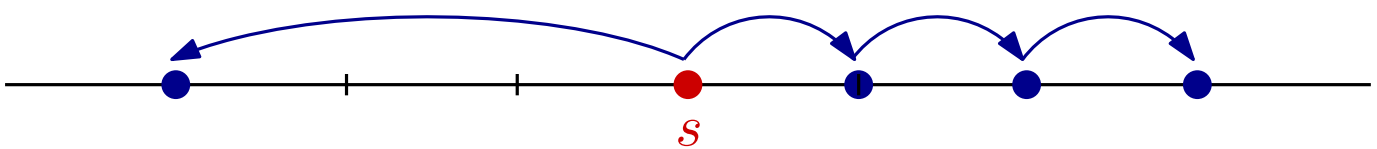


The dynamic broadcast range-assignment problem

Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

Example in $\mathbb{R}^1$ (online algorithm that does not decrease any ranges)



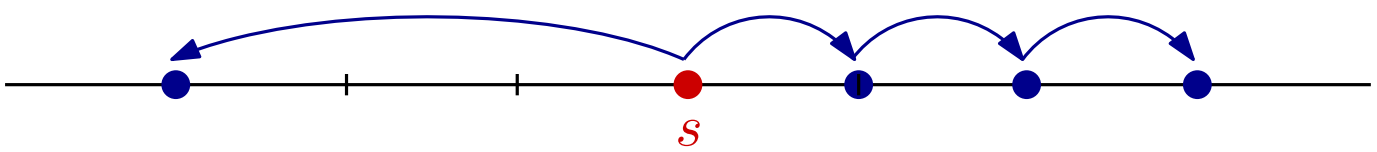
- cost $= 3^2 + 1^2 + 1^2 = 11$
- $\textsc{Opt} = 9$

The dynamic broadcast range-assignment problem

Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

Example in $\mathbb{R}^1$

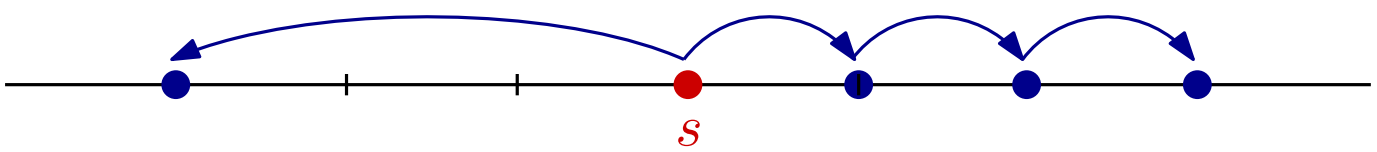


The dynamic broadcast range-assignment problem

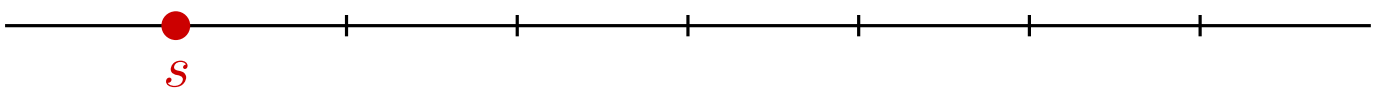
Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

Example in $\mathbb{R}^1$



online algorithm without recourse can have arbitrarily bad approximation ratio

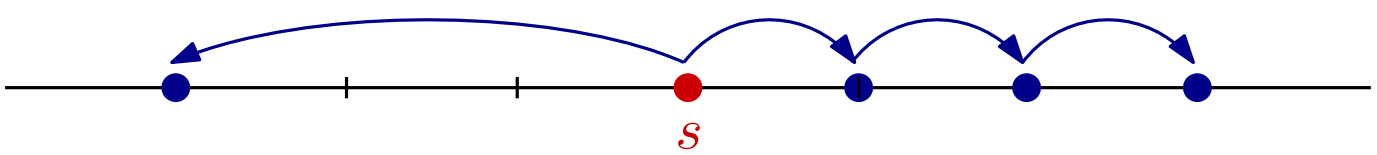


The dynamic broadcast range-assignment problem

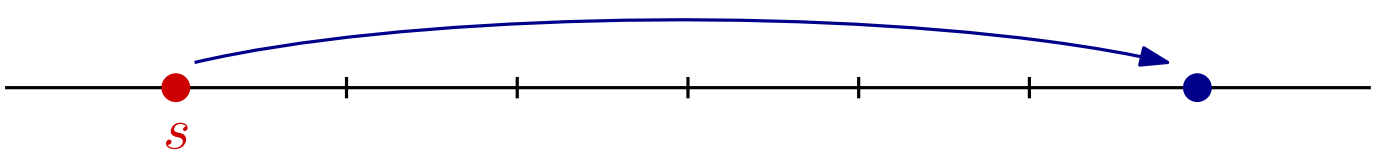
Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

Example in $\mathbb{R}^1$



online algorithm without recourse can have arbitrarily bad approximation ratio

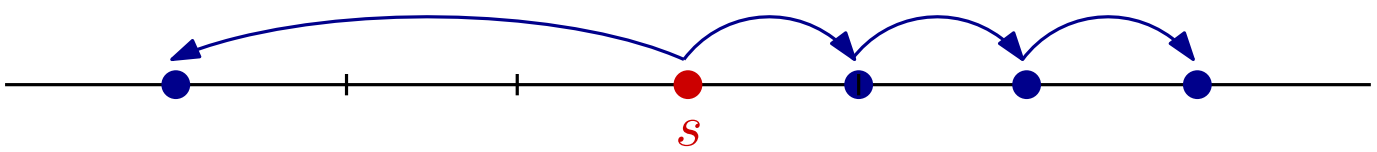


The dynamic broadcast range-assignment problem

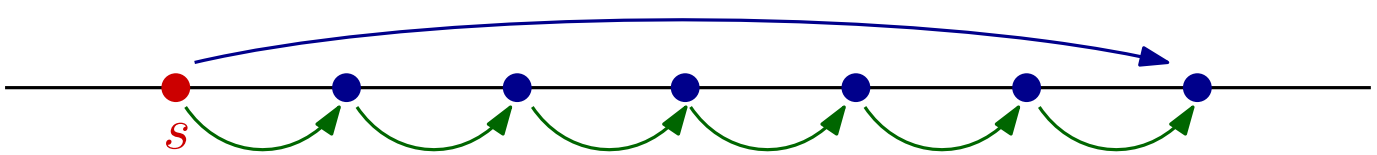
Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

Example in $\mathbb{R}^1$



online algorithm without recourse can have arbitrarily bad approximation ratio

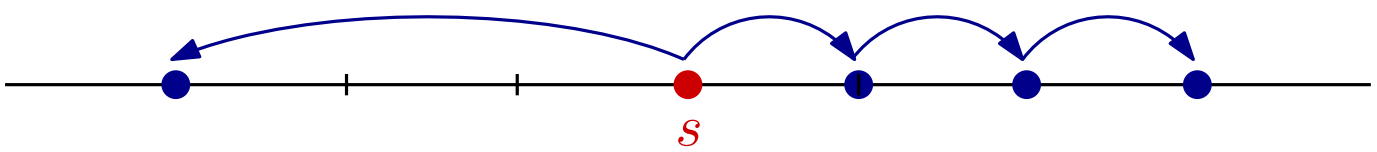


The dynamic broadcast range-assignment problem

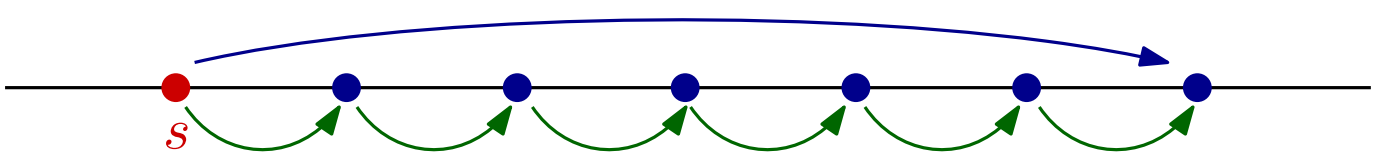
Dynamic version

- source is fixed
- points arrive one by one (and may also be deleted)
- when point arrives it initially gets range zero
- edit distance = number of points whose range changes

Example in $\mathbb{R}^1$



online algorithm without recourse can have arbitrarily bad approximation ratio



- $\text{OPT}_{\text{offline}} = n^2$ without recourse
- $\text{OPT} = n$

Stable algorithms for the dynamic broadcast range-assignment problem

Our results [dB-Sadhukhan-Spieksma, SWAT '22]

In $\mathbb{R}^1$

- there is a SAS with $k_\varepsilon = O(1/\varepsilon)$, which is optimal
- various algorithm with very small stability parameter
  - 1-stable $(6 + 2\sqrt{5})$-approximation algorithm
  - 2-stable 2-approximation algorithm
  - 3-stable 1.97 approximation algorithm

In $\mathbb{S}^1$

- structure of optimal solution is essentially the same as in $\mathbb{R}^1$
- SAS does not exist

In $\mathbb{R}^2$

- SAS does not exist
- 17-stable 12-approximation algorithm

Stable algorithms for the dynamic broadcast range-assignment problem

Our results [dB-Sadhukhan-Spieksma, SWAT '22]

In $\mathbb{R}^1$

- there is a SAS with $k_\varepsilon = O(1/\varepsilon)$, which is optimal
- various algorithm with very small stability parameter
  - 1-stable $(6 + 2\sqrt{5})$-approximation algorithm
  - 2-stable 2-approximation algorithm
  - 3-stable 1.97 approximation algorithm

In $\mathbb{S}^1$

- structure of optimal solution is essentially the same as in $\mathbb{R}^1$
- SAS does not exist

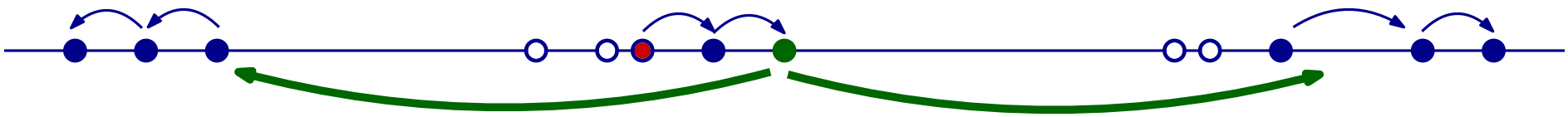
In $\mathbb{R}^2$

- SAS does not exist
- 17-stable 12-approximation algorithm

A Stable Approximation Scheme (SAS) in $\mathbb{R}^1$

Theorem. There is always a broadcast tree with the following structure.

[Caragiannis-Kaklamanis-Kanellopoulos ISAAC 2002]

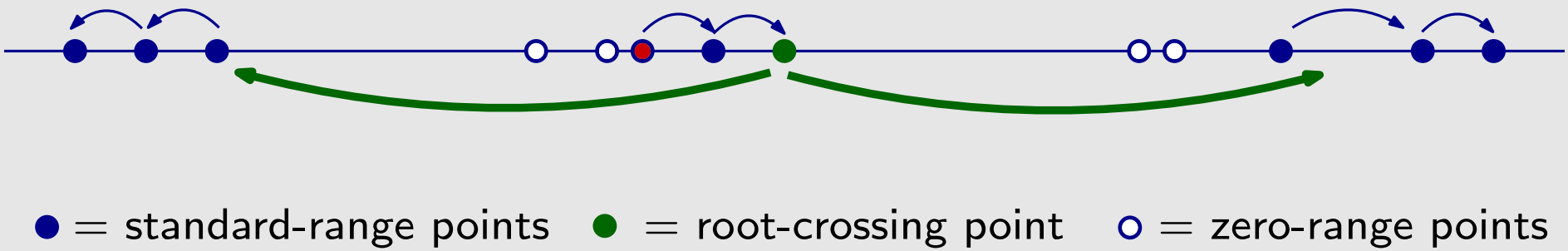


- ● = standard-range points
- ● = single root-crossing point
- ○ = zero-range points

Note: there can be more edges in the communication graph.

A Stable Approximation Scheme (SAS) in $\mathbb{R}^1$

structure of optimal solution

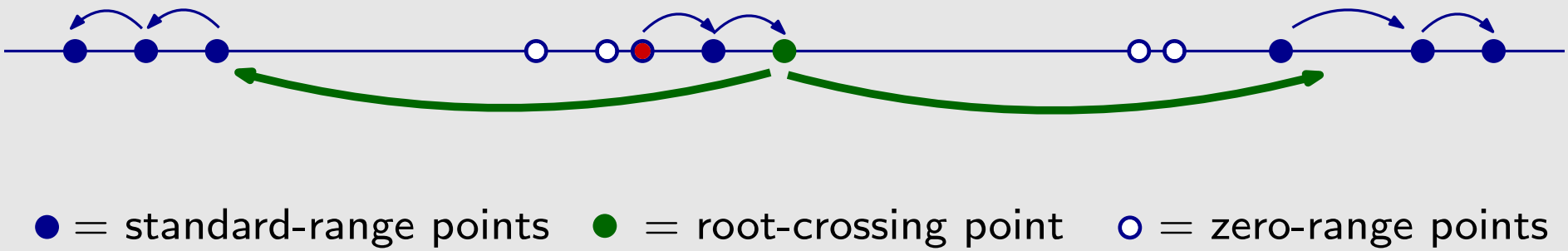


insertion/deletion:

root-crossing point can change
$\Longrightarrow$ many points can change from zero range to standard range (and vice versa)

A Stable Approximation Scheme (SAS) in $\mathbb{R}^1$

structure of optimal solution



insertion/deletion:

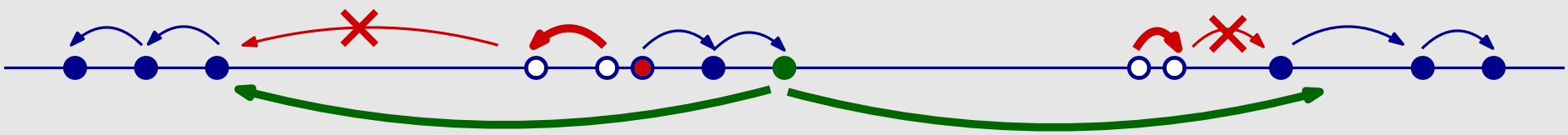
root-crossing point can change
$\Longrightarrow$ many points can change from zero range to standard range (and vice versa)

SAS: maintain solution with following property

- give zero-range points in optimal solution a standard range instead ...
- ... except the k zero-range points with largest standard range

A Stable Approximation Scheme (SAS) in $\mathbb{R}^1$

structure of optimal solution



● = standard-range points ● = root-crossing point ○ = zero-range points

insertion/deletion:

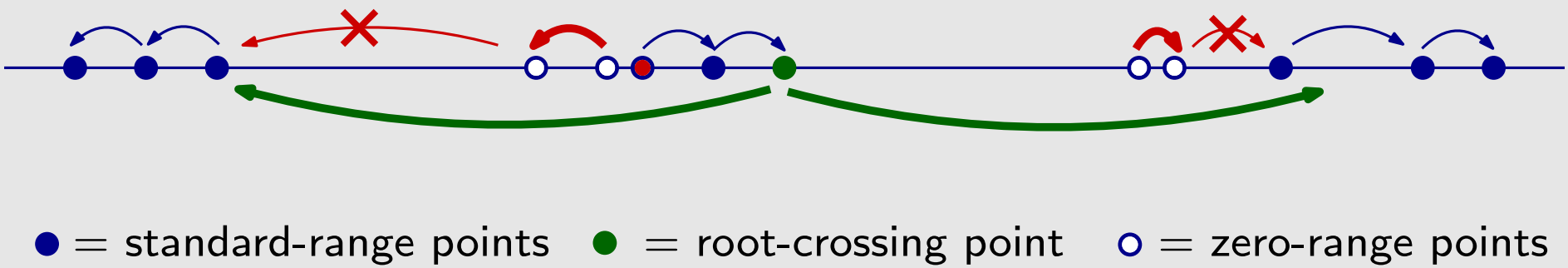
root-crossing point can change
$\Longrightarrow$ many points can change from zero range to standard range (and vice versa)

SAS: maintain solution with following property

- give zero-range points in optimal solution a standard range instead . . .
- . . . except the k zero-range points with largest standard range

A Stable Approximation Scheme (SAS) in $\mathbb{R}^1$

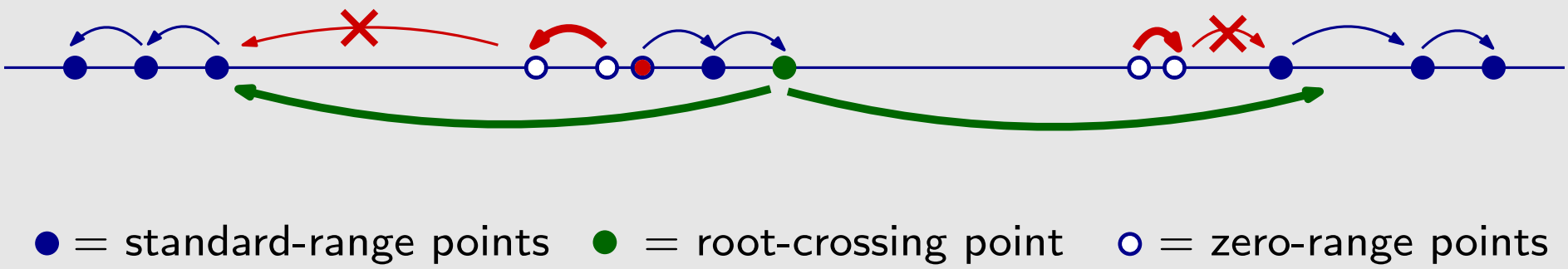
structure of optimal solution



Analysis of stability

A Stable Approximation Scheme (SAS) in $\mathbb{R}^1$

structure of optimal solution

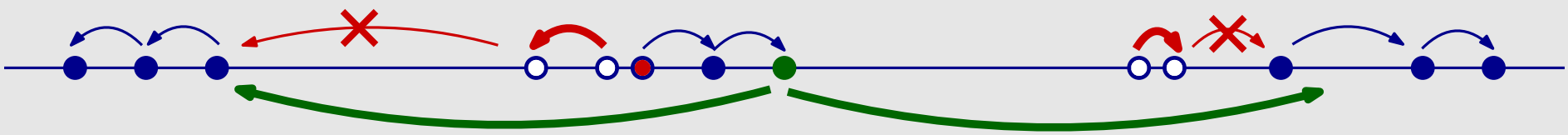


Analysis of stability

- root-crossing point can change: 2 ranges change
- k largest zero-range points may change: $2k$ ranges change
- new point gets range, standard range of predecessor changes: 2 ranges change
- source (=root) gets a different of its two standard ranges

A Stable Approximation Scheme (SAS) in $\mathbb{R}^1$

structure of optimal solution



● = standard-range points ● = root-crossing point ○ = zero-range points

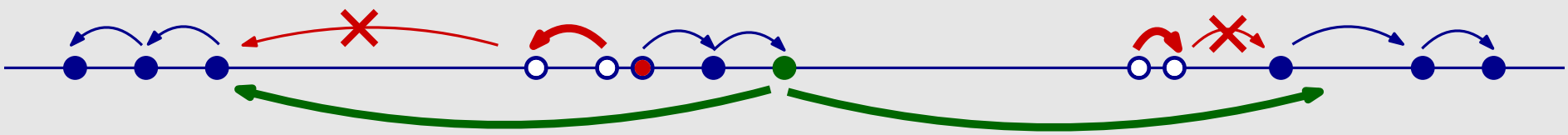
Analysis of stability

- root-crossing point can change: 2 ranges change
- k largest zero-range points may change: $2k$ ranges change
- new point gets range, standard range of predecessor changes: 2 ranges change
- source (=root) gets a different of its two standard ranges

Algorithm is $2k + 5$ stable

A Stable Approximation Scheme (SAS) in $\mathbb{R}^1$

structure of optimal solution

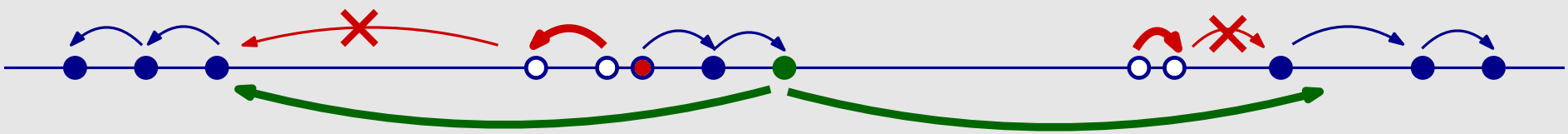


● = standard-range points ● = root-crossing point ○ = zero-range points

Analysis of approximation ratio (on one side of root-crossing point)

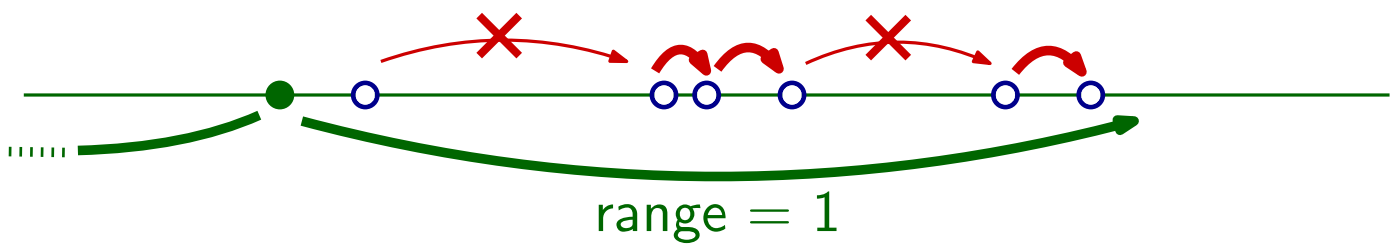
A Stable Approximation Scheme (SAS) in $\mathbb{R}^1$

structure of optimal solution



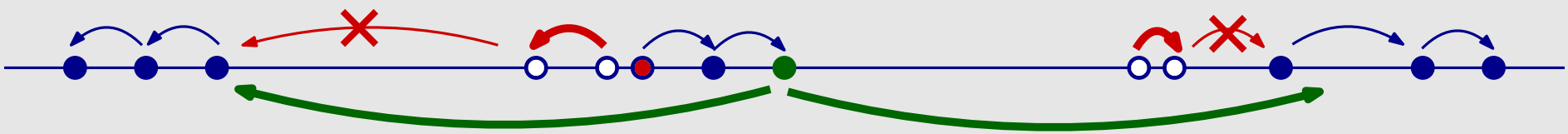
● = standard-range points ● = root-crossing point ○ = zero-range points

Analysis of approximation ratio (on one side of root-crossing point)



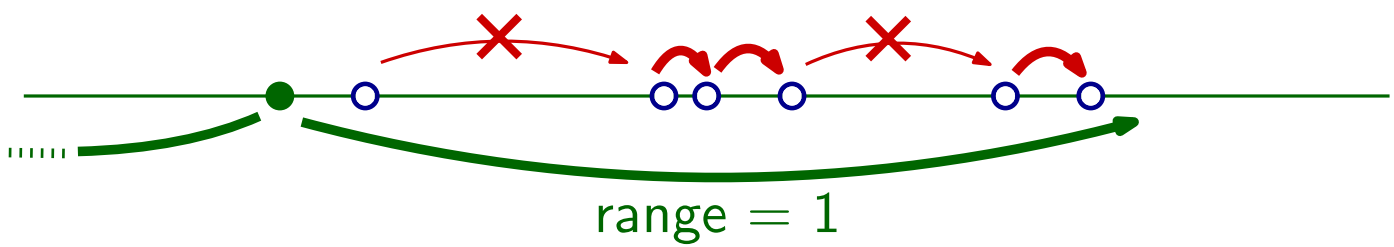
A Stable Approximation Scheme (SAS) in $\mathbb{R}^1$

structure of optimal solution



● = standard-range points ● = root-crossing point ○ = zero-range points

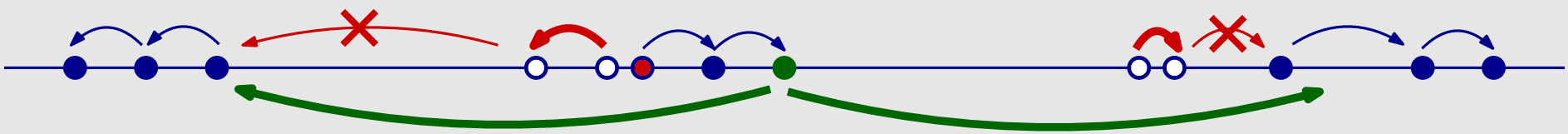
Analysis of approximation ratio (on one side of root-crossing point)



k largest zero-range points keep zero range
$\Longrightarrow$ other zero-range points have range $< 1/k$
$\Longrightarrow$ additional cost $< k \cdot (1/k)^2 = 1/k$

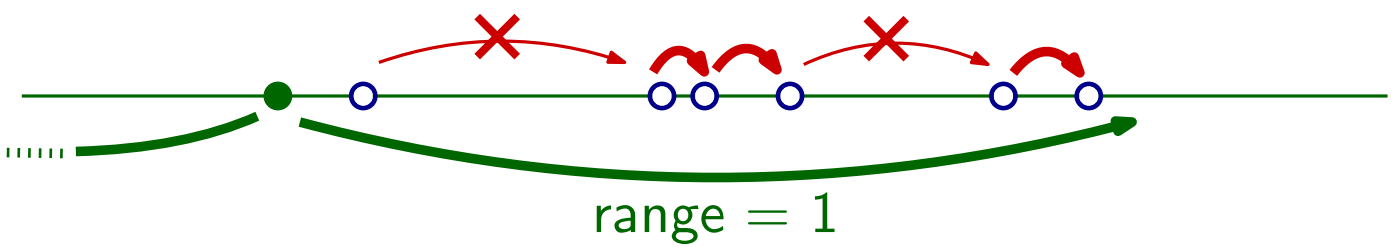
A Stable Approximation Scheme (SAS) in $\mathbb{R}^1$

structure of optimal solution



● = standard-range points ● = root-crossing point ○ = zero-range points

Analysis of approximation ratio (on one side of root-crossing point)



k largest zero-range points keep zero range
$\Longrightarrow$ other zero-range points have range $< 1/k$
$\Longrightarrow$ additional cost $< k \cdot (1/k)^2 = 1/k$

more precise analysis
approx ratio $= 1 + 4/k$

A Stable Approximation Scheme (SAS) in $\mathbb{R}^1$

- algorithm is $(2k+5)$-stable
- aproximation ratio is $1+4/k$

By picking a suitable $k = O(1/\varepsilon)$ we get:

Theorem. The broadcast range-assignment problem in $\mathbb{R}^1$ admits a SAS with stability parameter $k_\varepsilon = O(1/\varepsilon)$, which is optimal.

Stable algorithms for the dynamic broadcast range-assignment problem

Our results [dB-Sadhukhan-Spieksma, SWAT '22]

In $\mathbb{R}^1$

- there is a SAS with $k_\varepsilon = O(1/\varepsilon)$, which is optimal
- various algorithm with very samll stability parameter
  - 1-stable $(6 + 2\sqrt{5})$-approximation algorithm
  - 2-stable 2-approximation algorithm
  - 3-stable 1.97 approximation algorithm

In $\mathbb{S}^1$

- structure of optimal solution is essentially the same as in $\mathbb{R}^1$
- SAS does not exist

In $\mathbb{R}^2$

- SAS does not exist
- 17-stable 12-approximation algorithm

Stable algorithms for the dynamic broadcast range-assignment problem

Our results [dB-Sadhukhan-Spieksma, SWAT '22]

In $\mathbb{R}^1$

- there is a SAS with $k_\varepsilon = O(1/\varepsilon)$, which is optimal
- various algorithm with very samll stability parameter
  - 1-stable $(6 + 2\sqrt{5})$-approximation algorithm
  - 2-stable 2-approximation algorithm
  - 3-stable 1.97 approximation algorithm

In $\mathbb{S}^1$

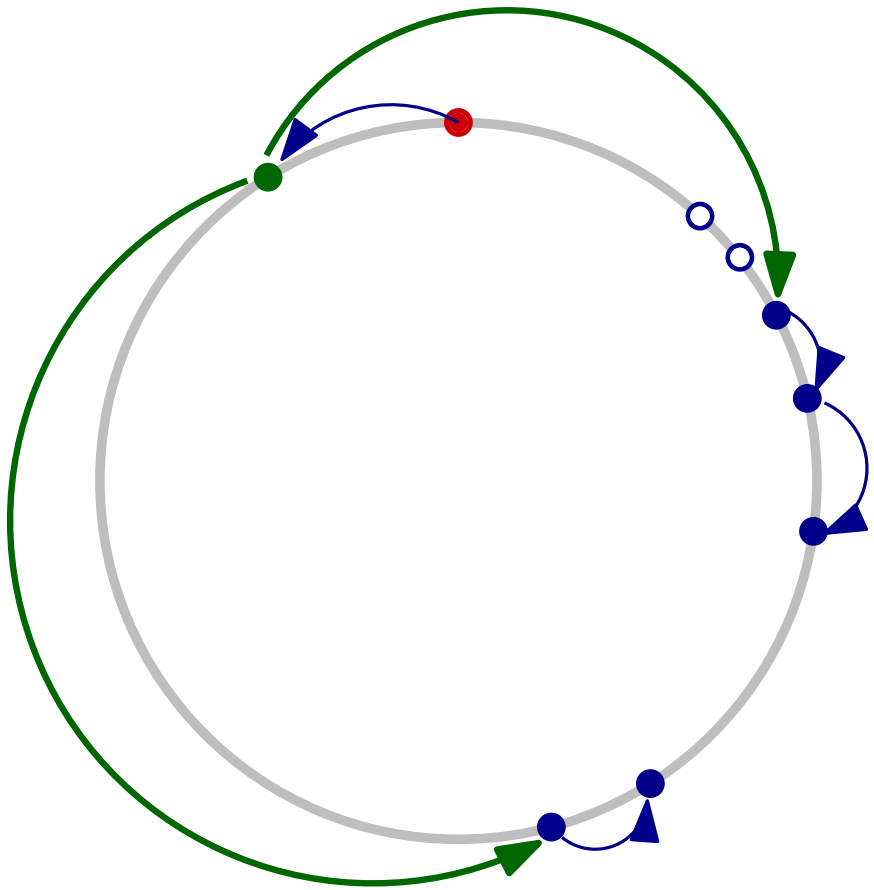
- structure of optimal solution is essentially the same as in $\mathbb{R}^1$
- SAS does not exist

In $\mathbb{R}^2$

- SAS does not exist
- 17-stable 12-approximation algorithm

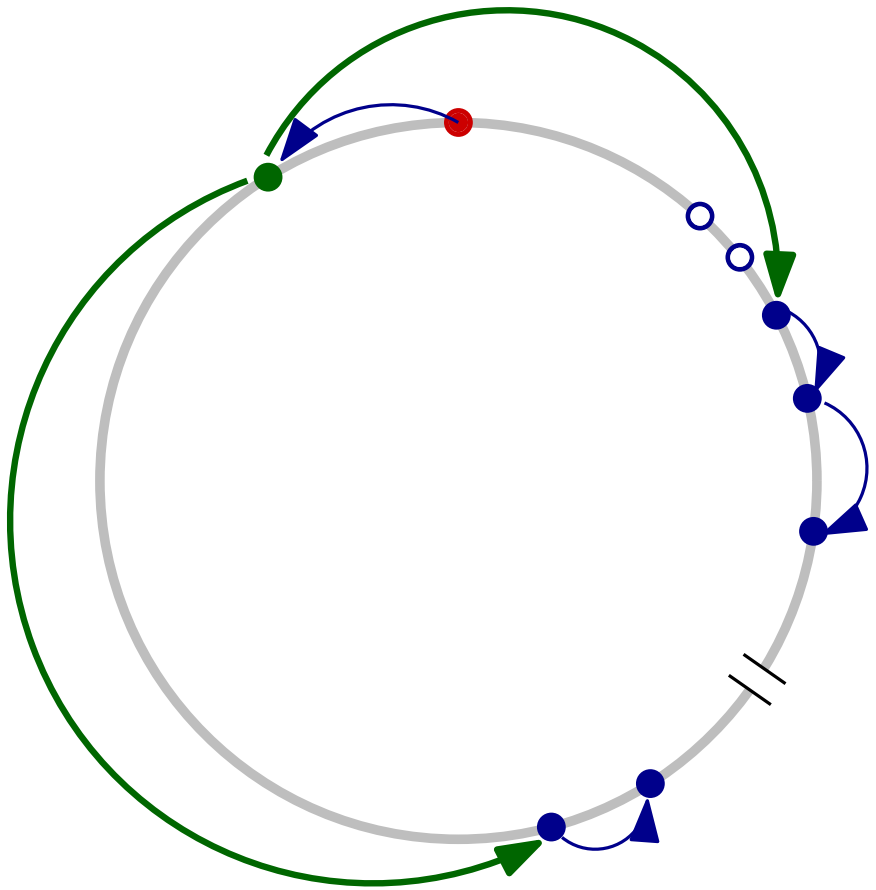
The structure of an optimal solution in $\mathbb{S}^1$

$\mathbb{S}^1$: 1-dimensional, circular space



The structure of an optimal solution in $\mathbb{S}^1$

$\mathbb{S}^1$: 1-dimensional, circular space



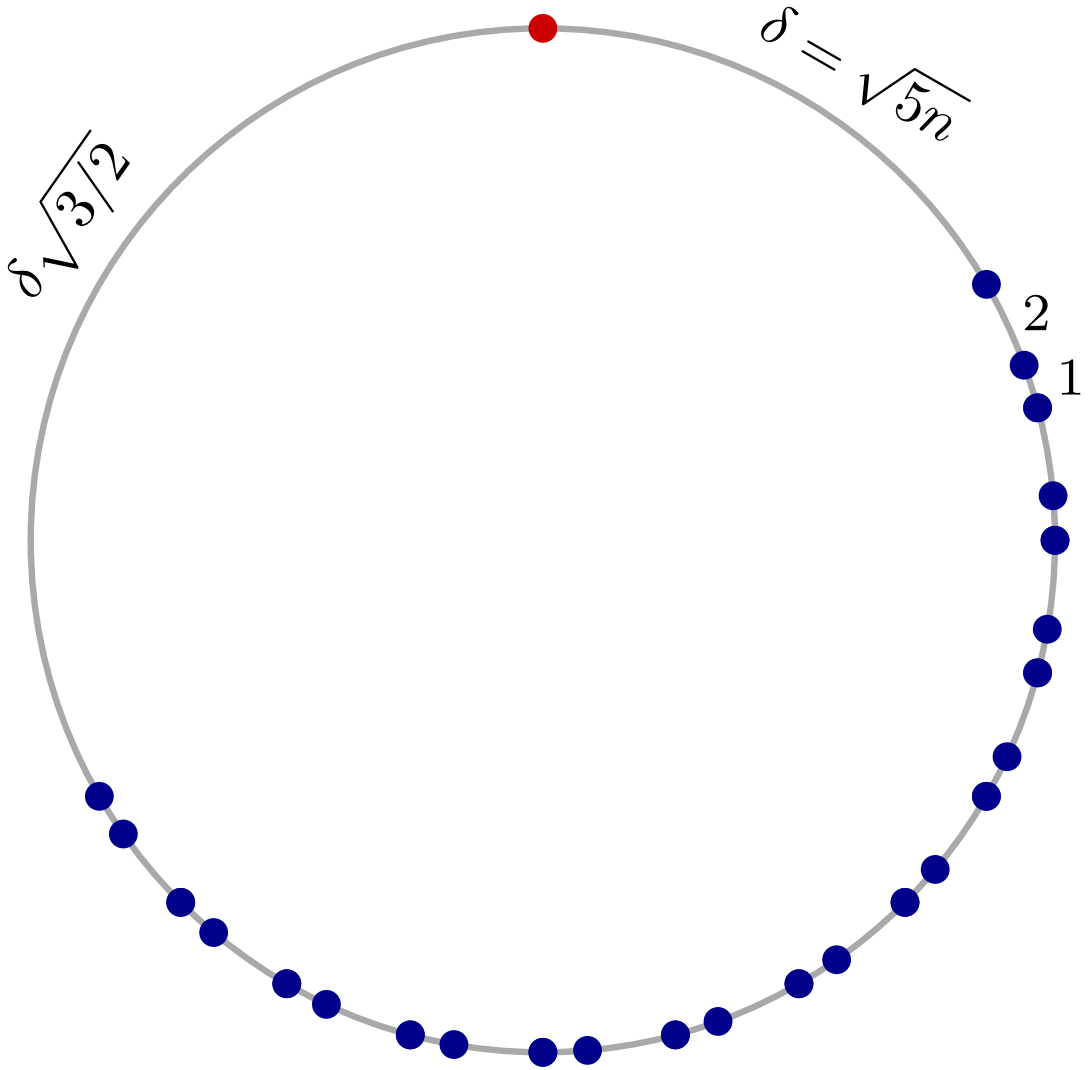
Theorem. There is an optimal solution that can be obtained by cutting the circle at an appropriate points, and then taking an optimal solution in $\mathbb{R}^1$.

No SAS in $\mathbb{S}^1$

Theorem. There is no SAS for the broadcast range-assignment problem in $\mathbb{S}^1$.

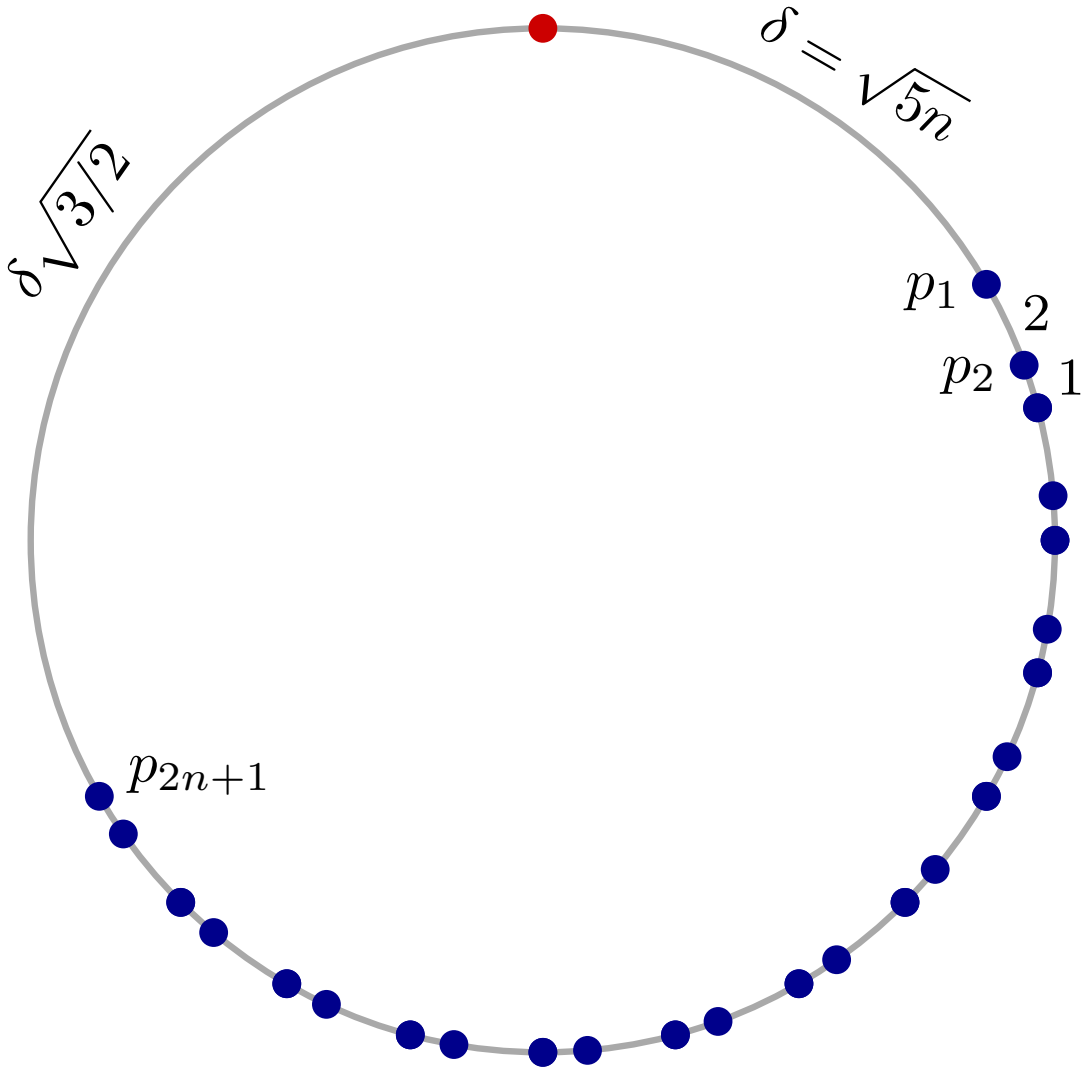
No SAS in $\mathbb{S}^1$

Theorem. There is no SAS for the broadcast range-assignment problem in $\mathbb{S}^1$.



No SAS in $\mathbb{S}^1$

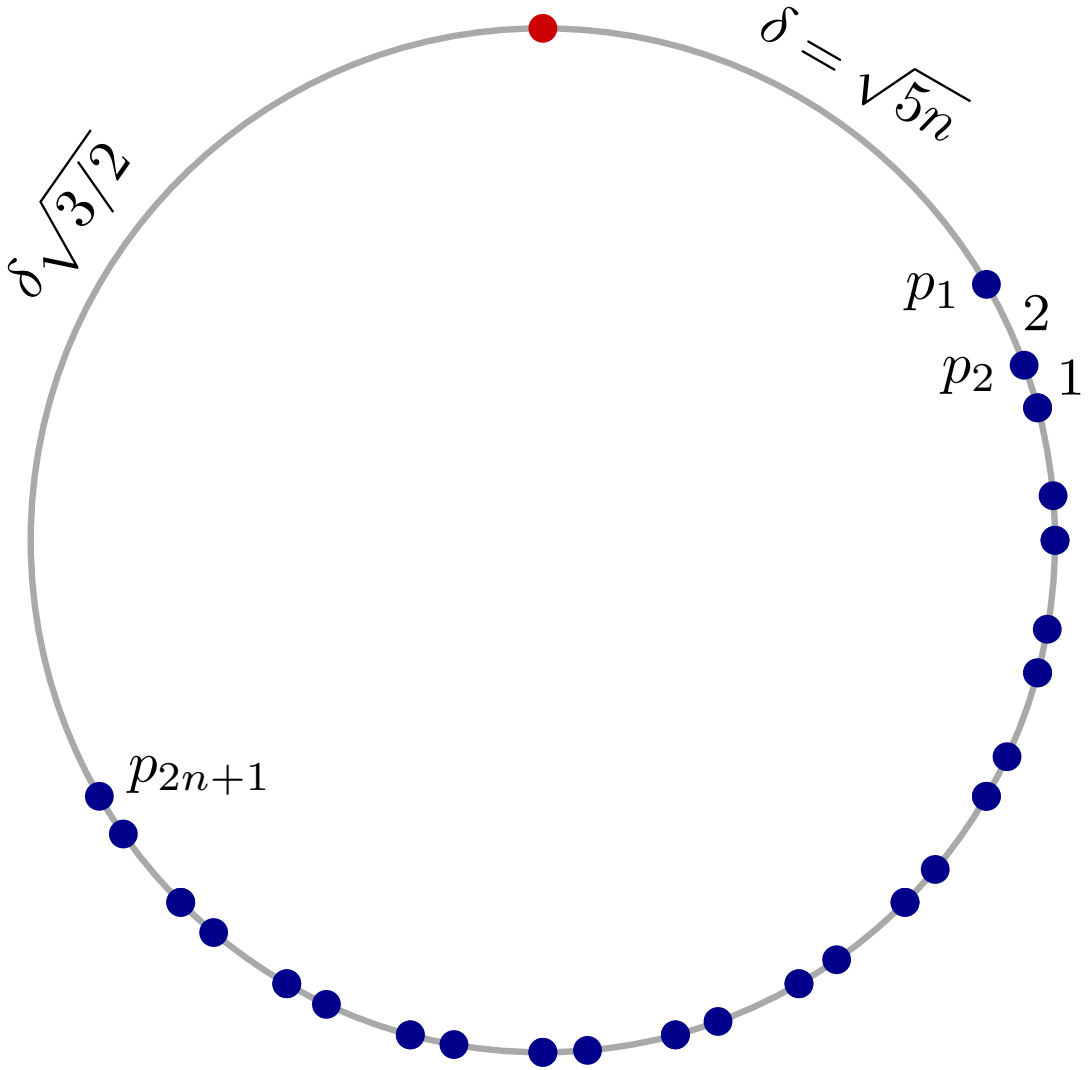
Theorem. There is no SAS for the broadcast range-assignment problem in $\mathbb{S}^1$.



$$\textsc{Opt} = \delta^2 + n(2^2 + 1^2) = 10n^2$$

No SAS in $\mathbb{S}^1$

Theorem. There is no SAS for the broadcast range-assignment problem in $\mathbb{S}^1$.



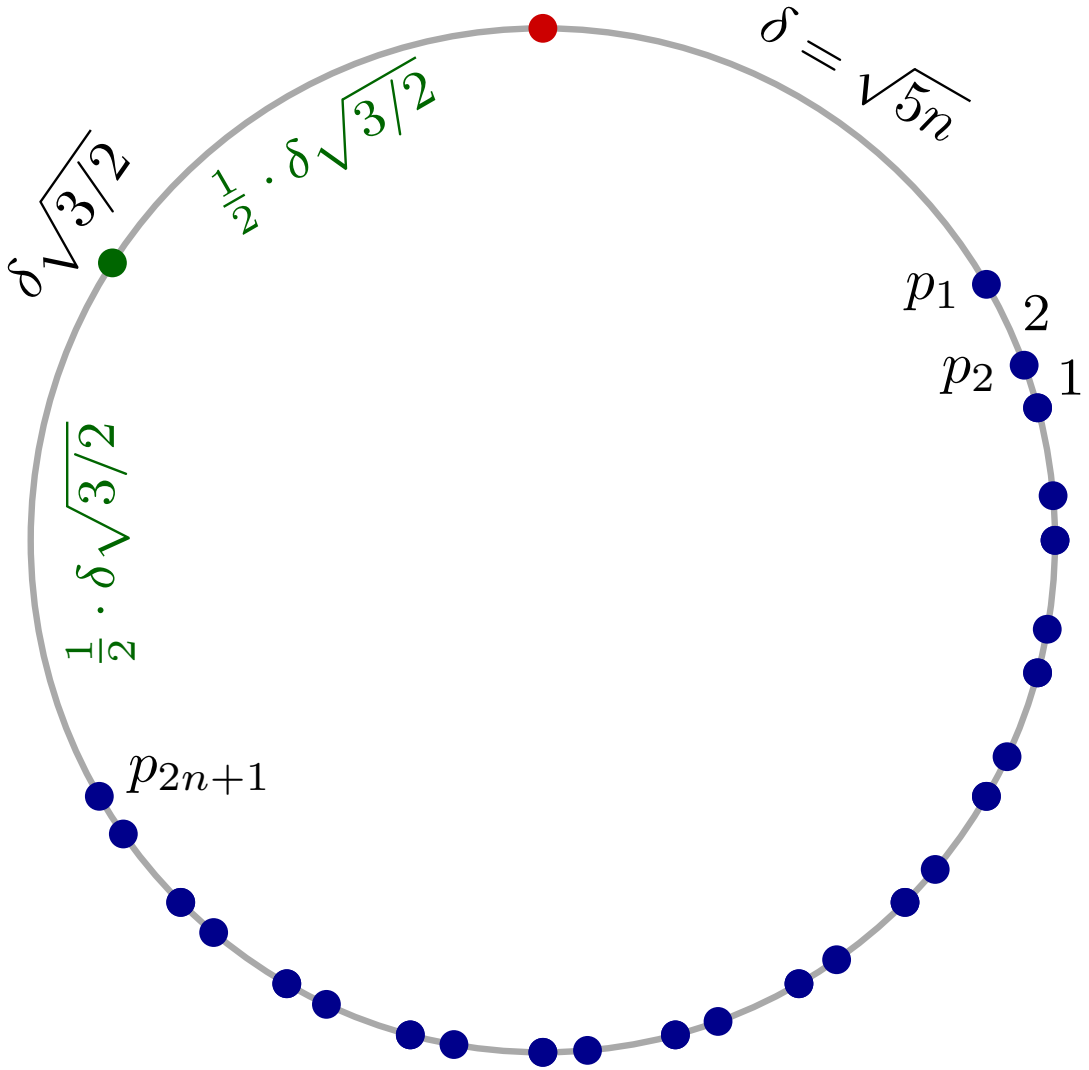
$$\textsc{Opt} = \delta^2 + n(2^2 + 1^2) = 10n^2$$

for $(1+\varepsilon)$-approximation

- p_{2n+1} cannot be reached with counter-clockwise arc
- most points p_i must have standard clockwise range

No SAS in $\mathbb{S}^1$

Theorem. There is no SAS for the broadcast range-assignment problem in $\mathbb{S}^1$.



$$\textsc{Opt} = \delta^2 + n(2^2 + 1^2) = 10n^2$$

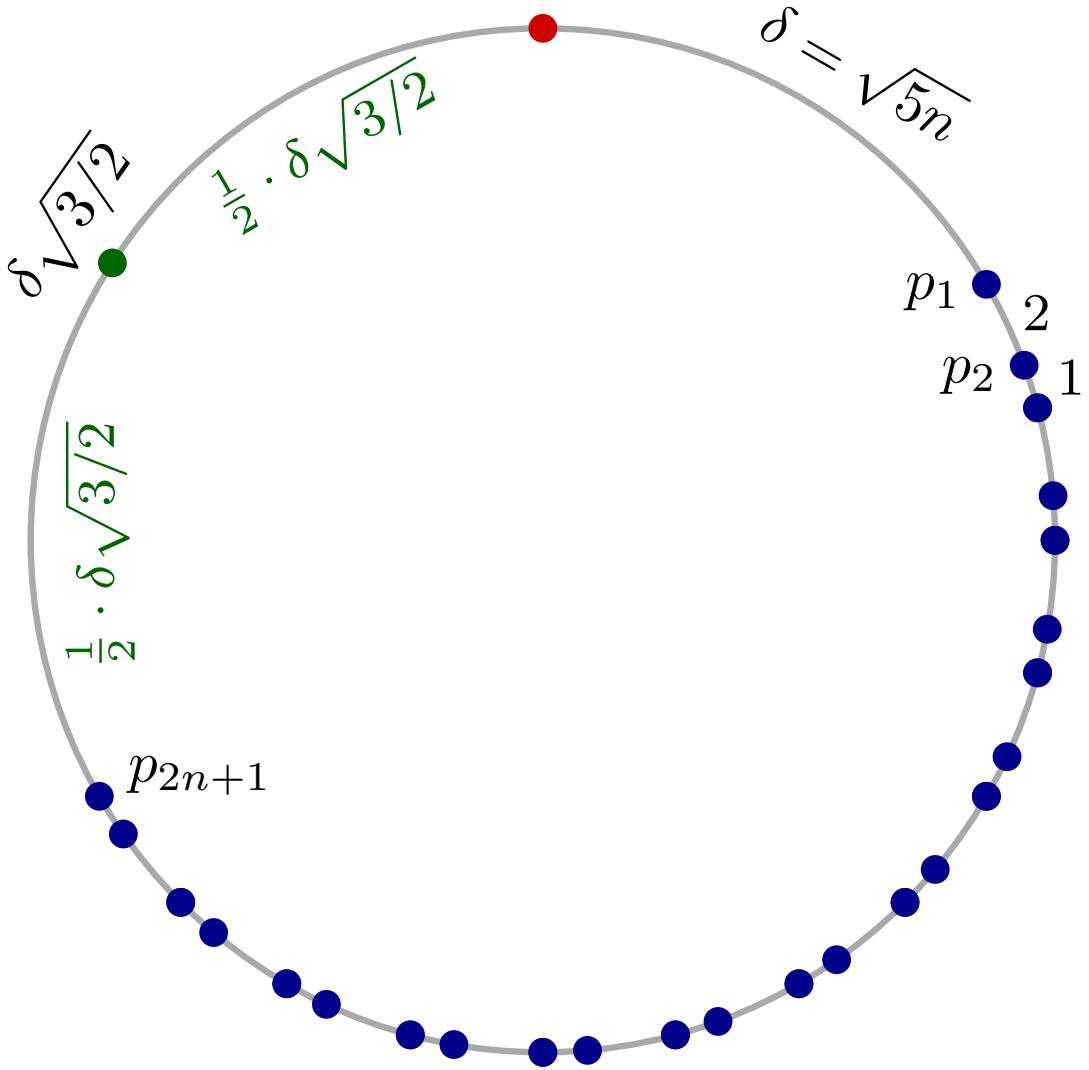
for $(1+\varepsilon)$-approximation

- p_{2n+1} cannot be reached with counter-clockwise arc
- most points p_i must have standard clockwise range

now point q arrives

No SAS in $\mathbb{S}^1$

Theorem. There is no SAS for the broadcast range-assignment problem in $\mathbb{S}^1$.



$$\textsc{Opt} = \delta^2 + n(2^2 + 1^2) = 10n^2$$

for $(1+\varepsilon)$-approximation

- p_{2n+1} cannot be reached with counter-clockwise arc
- most points p_i must have standard clockwise range

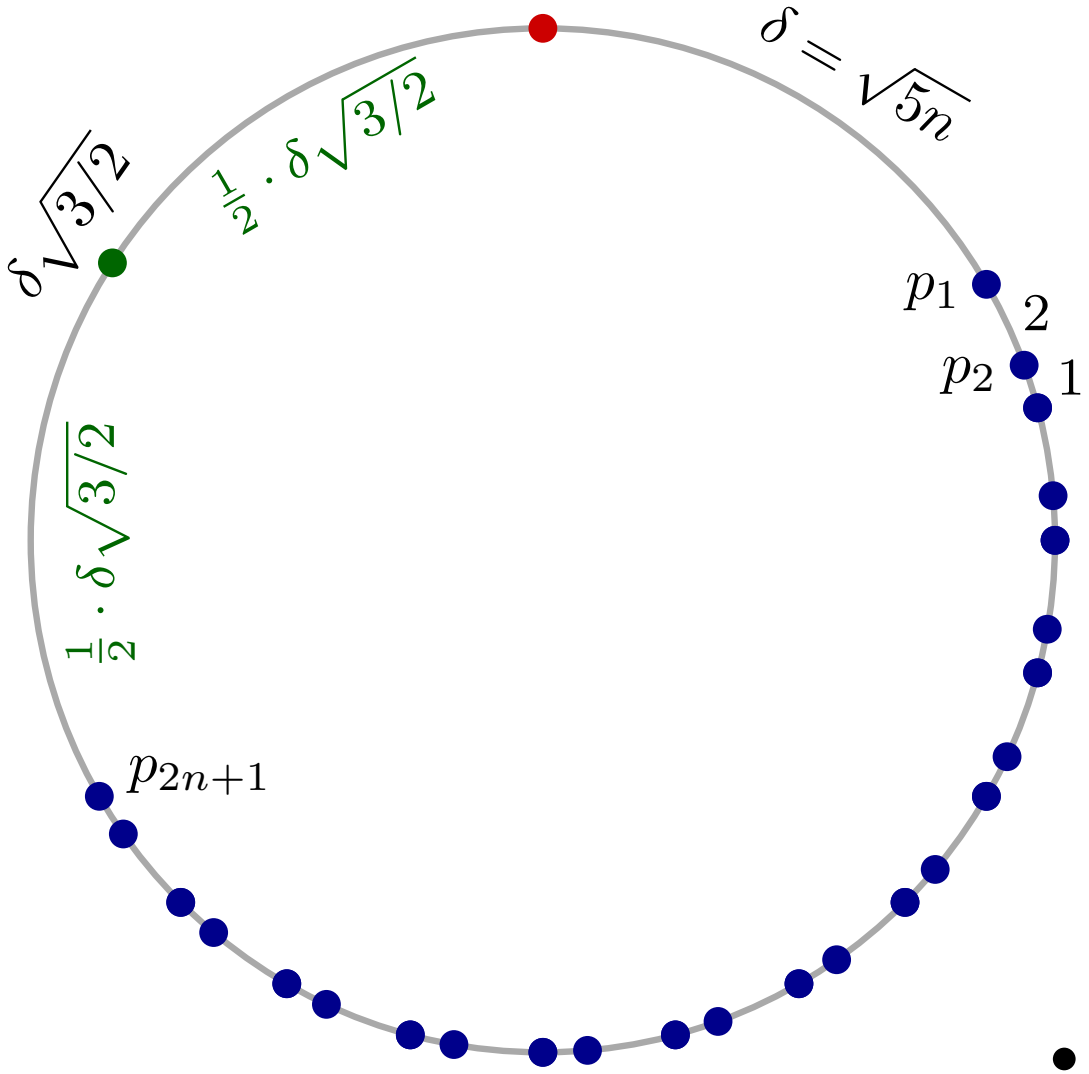
now point q arrives

new $\textsc{Opt}$

$$= 2(\tfrac{1}{2}(\delta\sqrt{3/2})^2 + n(2^2 + 1^2)$$
$$= 8\tfrac{3}{4}n$$

No SAS in $\mathbb{S}^1$

Theorem. There is no SAS for the broadcast range-assignment problem in $\mathbb{S}^1$.



$$\textsc{Opt} = \delta^2 + n(2^2 + 1^2) = 10n^2$$

for $(1+\varepsilon)$-approximation

- p_{2n+1} cannot be reached with counter-clockwise arc
- most points p_i must have standard clockwise range

now point q arrives

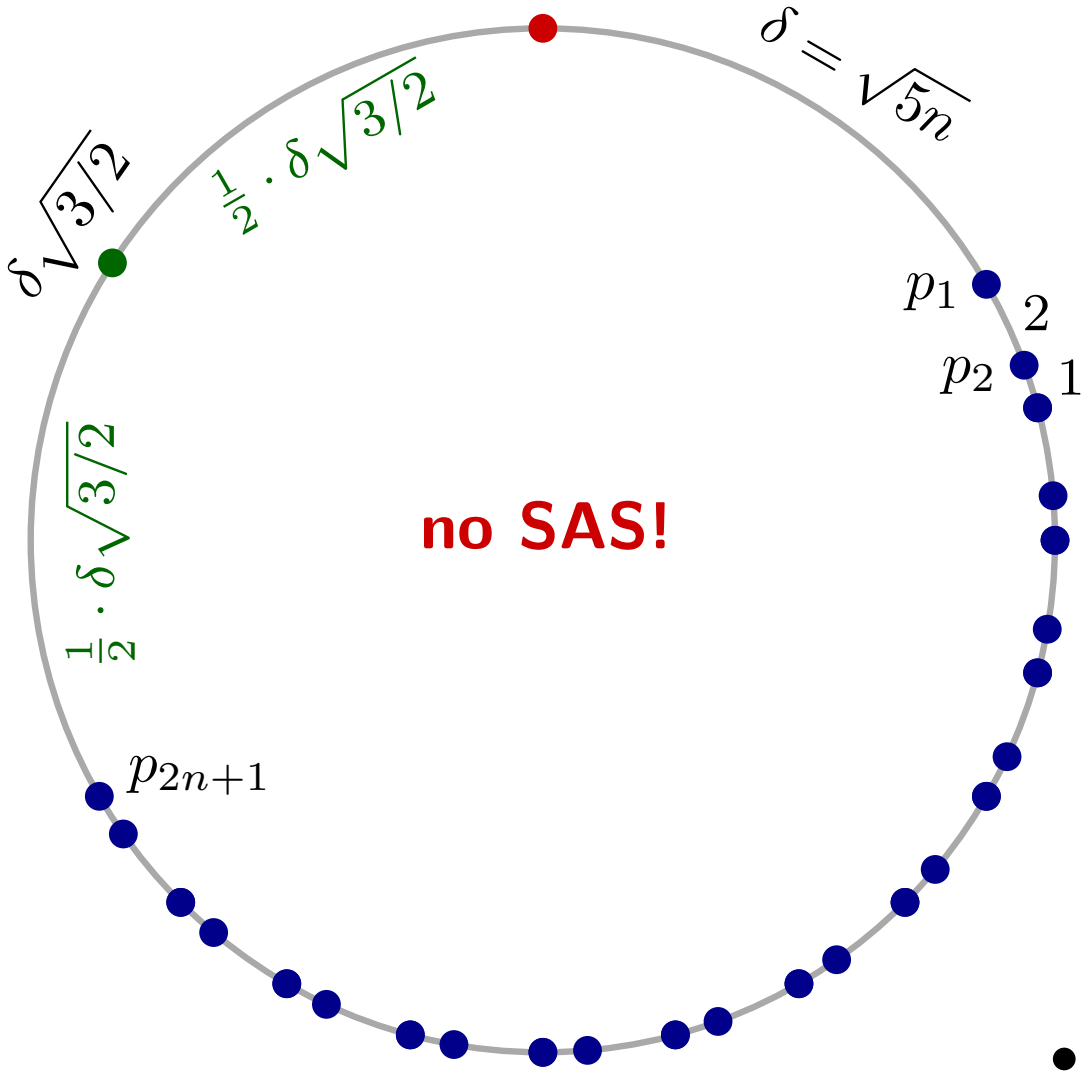
new $\textsc{Opt}$

$$= 2(\tfrac{1}{2}(\delta\sqrt{3/2})^2 + n(2^2 + 1^2)$$
$$= 8\tfrac{3}{4}n$$

- most points p_i must have standard counterclockwise range

No SAS in $\mathbb{S}^1$

Theorem. There is no SAS for the broadcast range-assignment problem in $\mathbb{S}^1$.



$$\textsc{Opt} = \delta^2 + n(2^2 + 1^2) = 10n^2$$

for $(1+\varepsilon)$-approximation

- p_{2n+1} cannot be reached with counter-clockwise arc
- most points p_i must have standard clockwise range

now point q arrives

new $\textsc{Opt}$
$= 2(\frac{1}{2}(\delta\sqrt{3/2})^2 + n(2^2 + 1^2)$
$= 8\frac{3}{4}n$

- most points p_i must have standard counterclockwise range

Stable algorithms for the dynamic broadcast range-assignment problem

Our results [dB-Sadhukhan-Spieksma, SWAT '22]

In $\mathbb{R}^1$

- there is a SAS with $k_\varepsilon = O(1/\varepsilon)$, which is optimal
- various algorithm with very samll stability parameter
  - 1-stable $(6 + 2\sqrt{5})$-approximation algorithm
  - 2-stable 2-approximation algorithm
  - 3-stable 1.97 approximation algorithm

In $\mathbb{S}^1$

- structure of optimal solution is essentially the same as in $\mathbb{R}^1$
- SAS does not exist

In $\mathbb{R}^2$

- SAS does not exist
- 17-stable 12-approximation algorithm

Stable algorithms for the dynamic broadcast range-assignment problem

Our results [dB-Sadhukhan-Spieksma, SWAT '22]

In $\mathbb{R}^1$

- there is a SAS with $k_\varepsilon = O(1/\varepsilon)$, which is optimal
- various algorithm with very samll stability parameter
  - 1-stable $(6 + 2\sqrt{5})$-approximation algorithm
  - 2-stable 2-approximation algorithm
  - 3-stable 1.97 approximation algorithm

In $\mathbb{S}^1$

- structure of optimal solution is essentially the same as in $\mathbb{R}^1$
- SAS does not exist

In $\mathbb{R}^2$

- SAS does not exist
- 17-stable 12-approximation algorithm

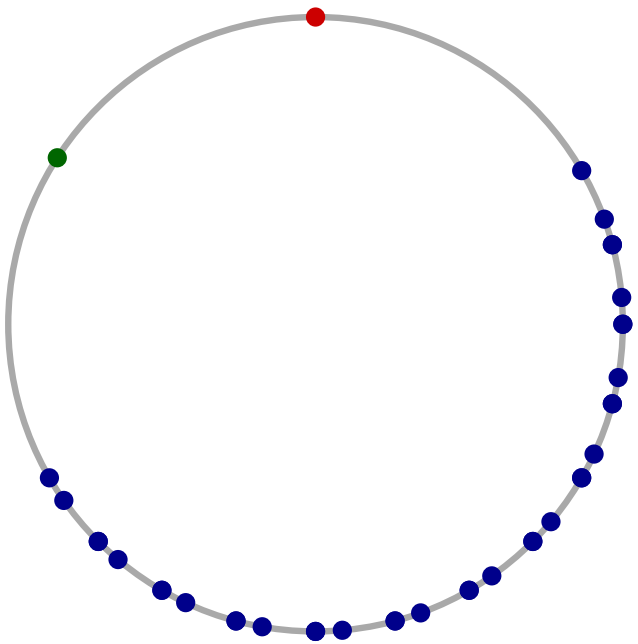
No SAS in $\mathbb{R}^2$

Theorem. There is no SAS for the broadcast range-assignment problem in $\mathbb{R}^2$.

No SAS in $\mathbb{R}^2$

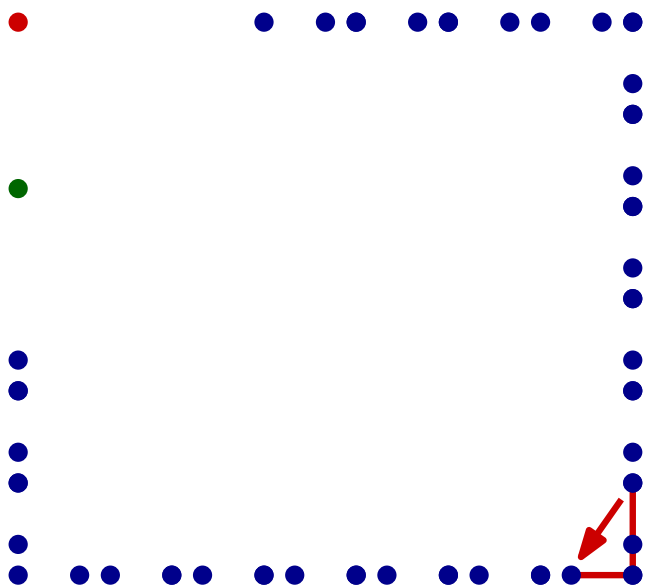
Theorem. There is no SAS for the broadcast range-assignment problem in $\mathbb{R}^2$.

construction in $\mathbb{S}^1$



$\Longrightarrow$

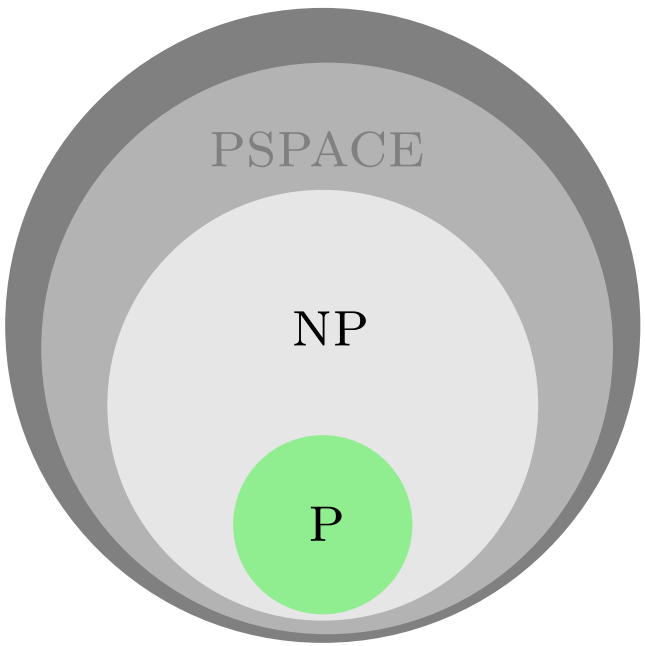
construction in $\mathbb{R}^2$



cutting corners does not help (enough)

Commercial break

Hardness: A Traditional Algorithmic View



Assumption: $\mathrm{P} \neq \mathrm{NP}$

- P = tractable
- NP-hard = untractable

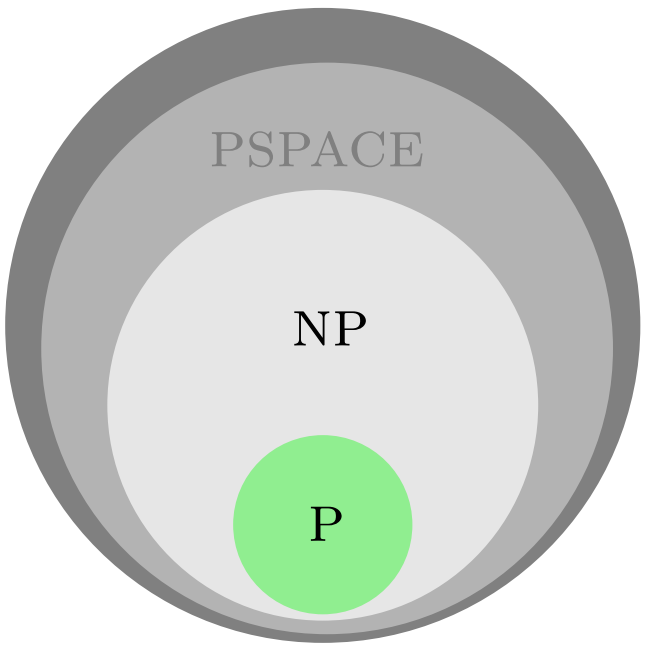
Algorithmic approach to problem solving

- Determine NP-hardness by reduction from known NP-hard problem
- give approximation algorithm

or

- try to give fastest possible polynomial-time algorithm
- sometimes lower bounds

A More Refined View



Assumption: $\mathrm{P} \neq \mathrm{NP}$

- P = tractable
- NP-hard = untractable

Algorithmic approach to problem solving

- Determine NP-hardness by reduction from known NP-hard problem
- give approximation algorithm
- study parameterized complexity
- study subexponential algorithms

or

- try to give fastest possible polynomial-time algorithm
- sometimes lower bounds

and relate problems to each other through conditional lower bounds

Subexponential algorithms

Subexponential algorithms: running time $2^{o(n)}$

- Typical examples: $2^{O(\sqrt{n})}$ or $n^{O(\sqrt{n})}$ or $2^{O(n/\log n)}$

Subexponential algorithms

Subexponential algorithms: running time $2^{o(n)}$

- Typical examples: $2^{O(\sqrt{n})}$ or $n^{O(\sqrt{n})}$ or $2^{O(n/\log n)}$

Do NP-hard problems have subexponential algorithms?

- 3-SAT: Is given 3-CNF formula witn n variables satisfiable?

$$(x_1 \vee \overline{x_2} \vee x_5) \wedge (x_3 \vee x_4 \vee \overline{x_6}) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_5})$$

Subexponential algorithms

Subexponential algorithms: running time $2^{o(n)}$

- Typical examples: $2^{O(\sqrt{n})}$ or $n^{O(\sqrt{n})}$ or $2^{O(n/\log n)}$

Do NP-hard problems have subexponential algorithms?

- 3-SAT: Is given 3-CNF formula witn n variables satisfiable?

$$(x_1 \vee \overline{x_2} \vee x_5) \wedge (x_3 \vee x_4 \vee \overline{x_6}) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_5})$$

Exponential-Time Hypothesis (ETH)

(Impagliazzo, Paturi and Zane 2001)

There is no sub-exponential algorithm for 3-SAT, that is, 3-SAT cannot be solved in $2^{o(n)}$ time.

What about other NP-hard problems?

Three classic NP-hard graph problems

Input: graph $G = (V, E)$ with n vertices

Independent Set

- Find largest set $I \subseteq V$ such that no two vertices in I are connected by an edge.

Vertex Cover

- Find smallest set $C \subseteq V$ that contains at least one endpoint of every edge in E.

Dominating Set

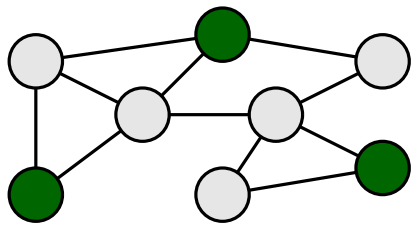
- Find smallest $D \subseteq V$ such that for each $v \in V$ either v or one of its neighbors is in D.

Three classic NP-hard graph problems

Input: graph $G = (V, E)$ with n vertices

Independent Set

- Find largest set $I \subseteq V$ such that no two vertices in I are connected by an edge.



Vertex Cover

- Find smallest set $C \subseteq V$ that contains at least one endpoint of every edge in E.

Dominating Set

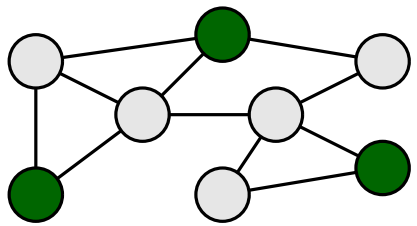
- Find smallest $D \subseteq V$ such that for each $v \in V$ either v or one of its neighbors is in D.

Three classic NP-hard graph problems

Input: graph $G = (V, E)$ with n vertices

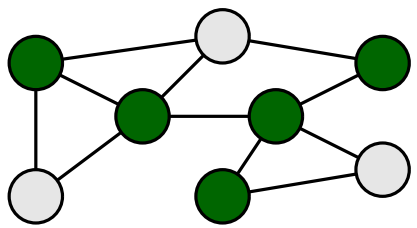
Independent Set

- Find largest set $I \subseteq V$ such that no two vertices in I are connected by an edge.



Vertex Cover

- Find smallest set $C \subseteq V$ that contains at least one endpoint of every edge in E.



Dominating Set

- Find smallest $D \subseteq V$ such that for each $v \in V$ either v or one of its neighbors is in D.

Three classic NP-hard graph problems

Input: graph $G = (V, E)$ with n vertices

Independent Set

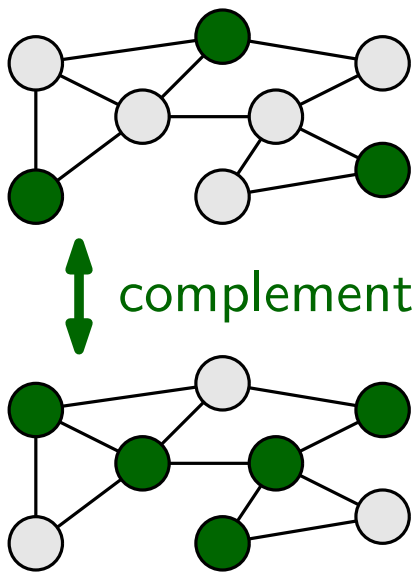
- Find largest set $I \subseteq V$ such that no two vertices in I are connected by an edge.

Vertex Cover

- Find smallest set $C \subseteq V$ that contains at least one endpoint of every edge in E.

Dominating Set

- Find smallest $D \subseteq V$ such that for each $v \in V$ either v or one of its neighbors is in D.



Three classic NP-hard graph problems

Input: graph $G = (V, E)$ with n vertices

Independent Set

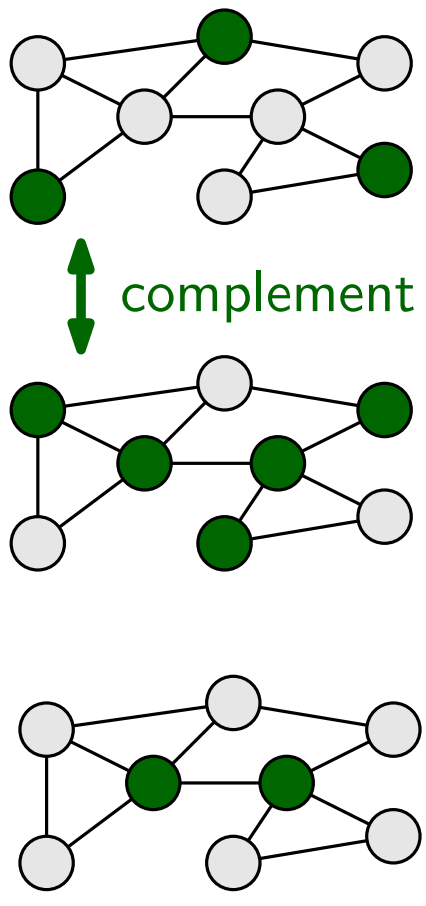
- Find largest set $I \subseteq V$ such that no two vertices in I are connected by an edge.

Vertex Cover

- Find smallest set $C \subseteq V$ that contains at least one endpoint of every edge in E.

Dominating Set

- Find smallest $D \subseteq V$ such that for each $v \in V$ either v or one of its neighbors is in D.



Three classic NP-hard graph problems

Are there subexponential algorithms for these classic graph problems?

Three classic NP-hard graph problems

Are there subexponential algorithms for these classic graph problems?

Theorem. Independent Set, Vertex Cover, and Dominating Set cannot be solved in $2^{o(n)}$ time, under ETH.

Three classic NP-hard graph problems

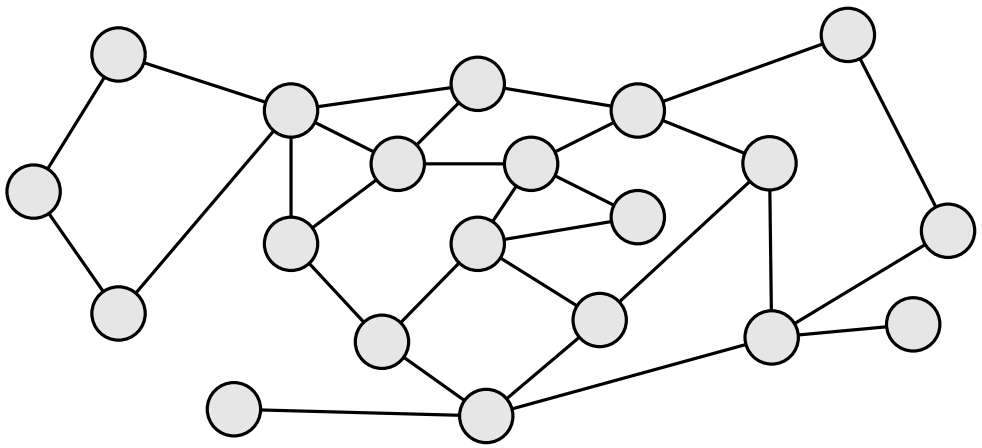
Are there subexponential algorithms for these classic graph problems?

Theorem. Independent Set, Vertex Cover, and Dominating Set cannot be solved in $2^{o(n)}$ time, under ETH.

but we can get subexponential algorithms on planar graphs

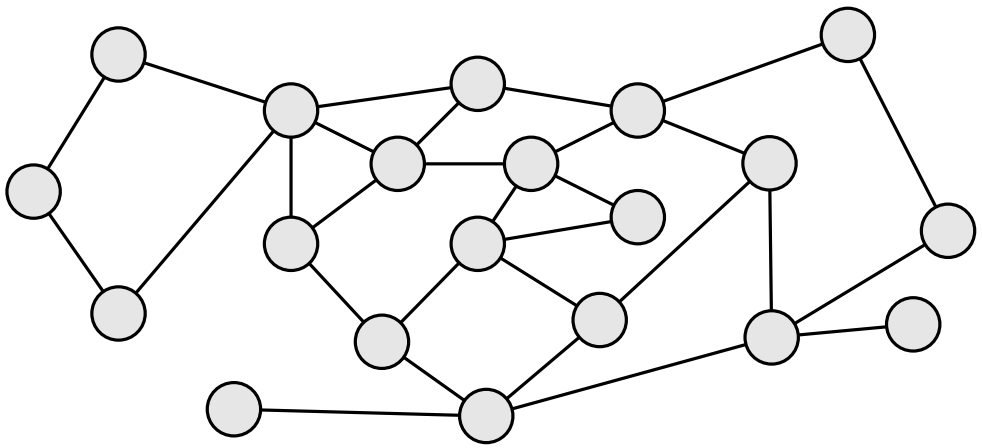
Planar graphs

Planar graphs: graphs that can be drawn without crossing edges



Planar graphs

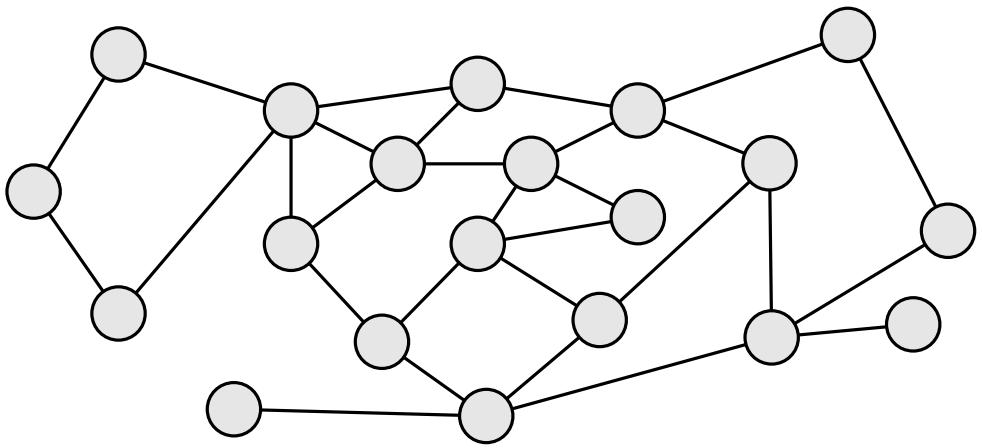
Planar graphs: graphs that can be drawn without crossing edges



Planar Separator Theorem (Lipton,Tarjan 1979)

Planar graphs

Planar graphs: graphs that can be drawn without crossing edges

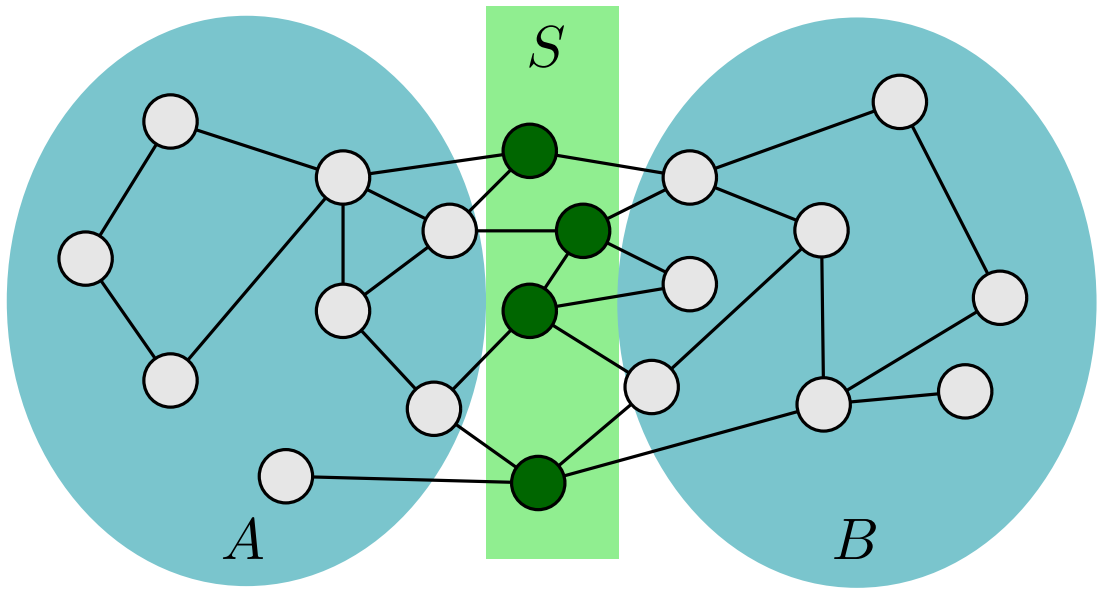


Planar Separator Theorem (Lipton,Tarjan 1979)

For any planar graph $G = (V, E)$ there is a separator $S \subset V$ of size $O(\sqrt{n})$ such that $V \setminus S$ can be partitioned into subsets A and B, each of size at most $\frac{2}{3}n$ and with no edges between them.

Planar graphs

Planar graphs: graphs that can be drawn without crossing edges

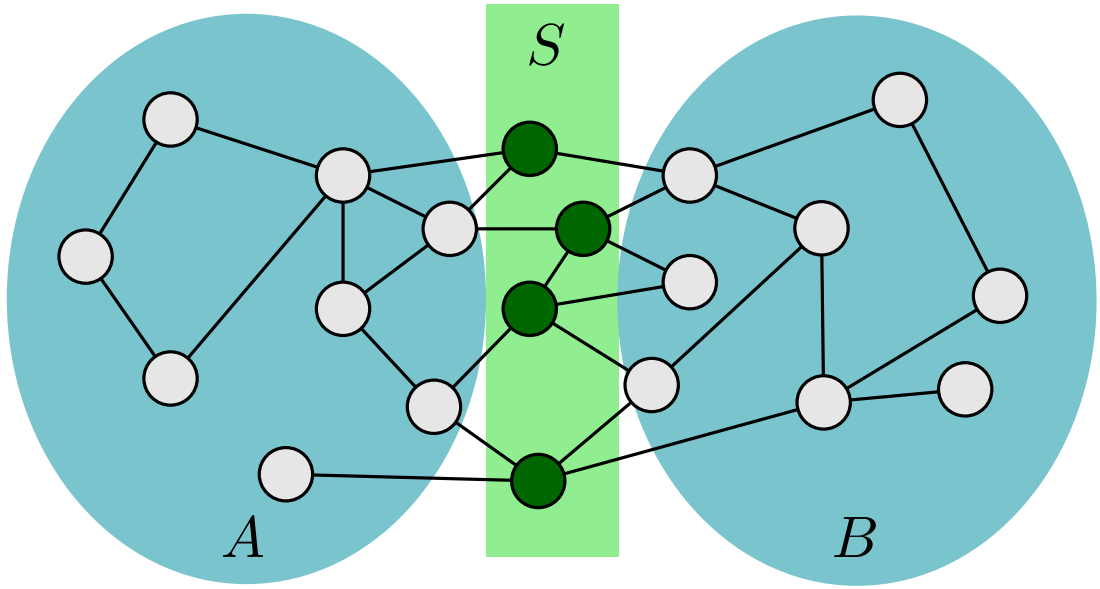


Planar Separator Theorem (Lipton,Tarjan 1979)

For any planar graph $G = (V, E)$ there is a separator $S \subset V$ of size $O(\sqrt{n})$ such that $V \setminus S$ can be partitioned into subsets A and B, each of size at most $\frac{2}{3}n$ and with no edges between them.

Planar graphs

Planar graphs: graphs that can be drawn without crossing edges



Planar Separator Theorem (Lipton,Tarjan 1979)

For any planar graph $G = (V, E)$ there is a separator $S \subset V$ of size $O(\sqrt{n})$ such that $V \setminus S$ can be partitioned into subsets A and B, each of size at most $\frac{2}{3}n$ and with no edges between them.

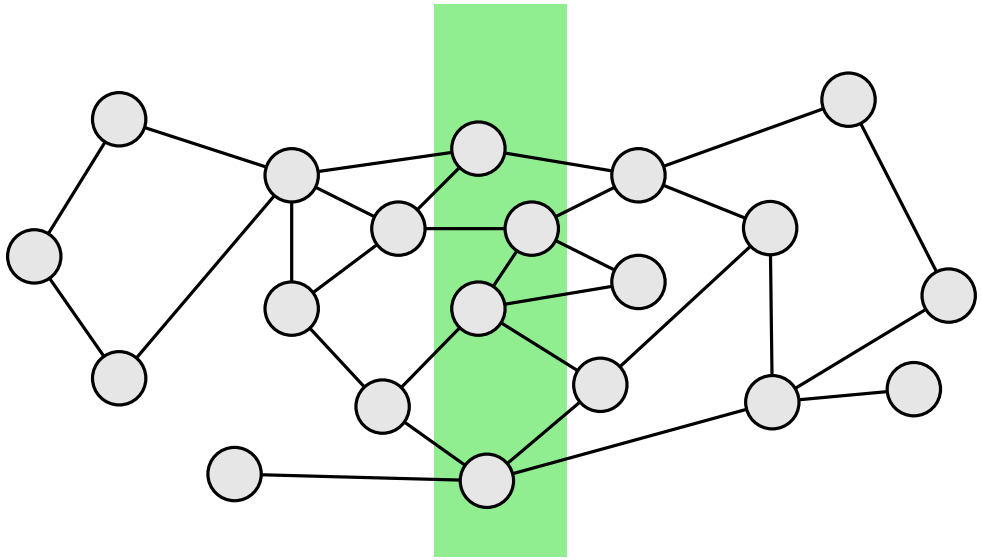
Such a $(2/3)$-balanced separator can be computed in $O(n)$ time.

Subexponential algorithms on planar graphs

Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.

Subexponential algorithms on planar graphs

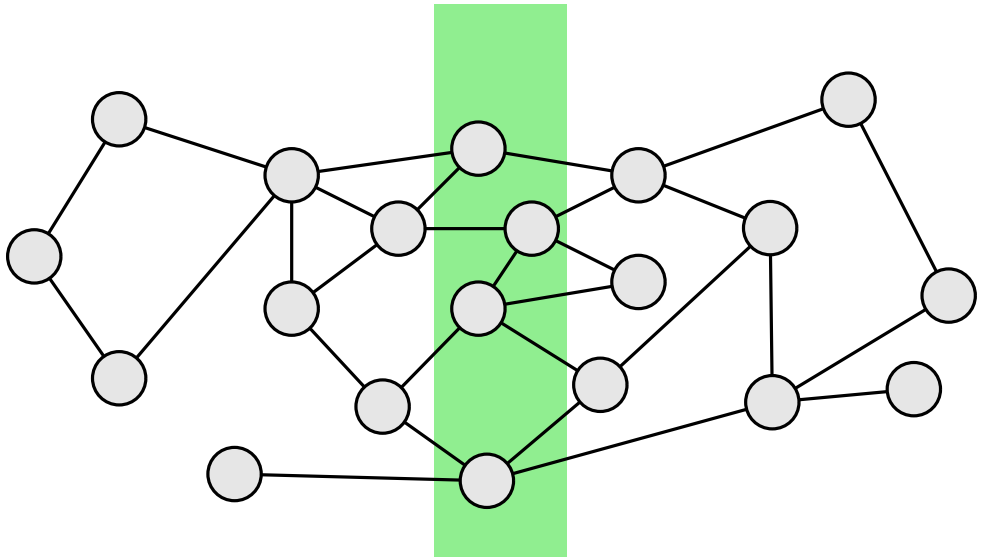
Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.



1. Compute $(2/3)$-balanced separator S of size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do
   (a) Recursively find max independent set I_A for $A \setminus \{\text{neighbors of } I_S\}$
   (b) Recursively find max independent set I_B for $B \setminus \{\text{neighbors of } I_S\}$
   (c) $I(S) := I_S \cup I_A \cup I_B$
3. Return the largest of the sets $I(S)$ found in Step 2.

Subexponential algorithms on planar graphs

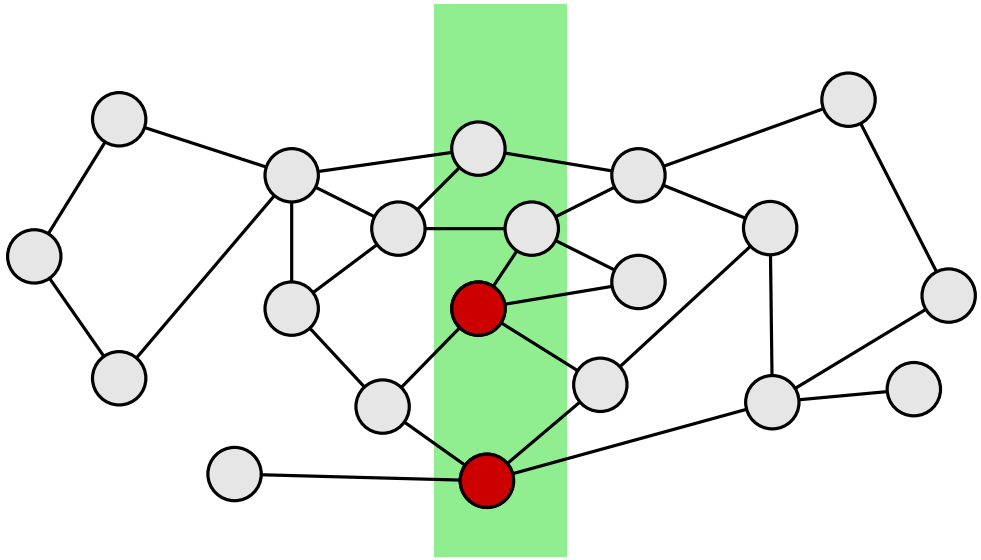
Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.



1. Compute $(2/3)$-balanced separator S of size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do
   (a) Recursively find max independent set I_A for $A \setminus \{\text{neighbors of } I_S\}$
   (b) Recursively find max independent set I_B for $B \setminus \{\text{neighbors of } I_S\}$
   (c) $I(S) := I_S \cup I_A \cup I_B$
3. Return the largest of the sets $I(S)$ found in Step 2.

Subexponential algorithms on planar graphs

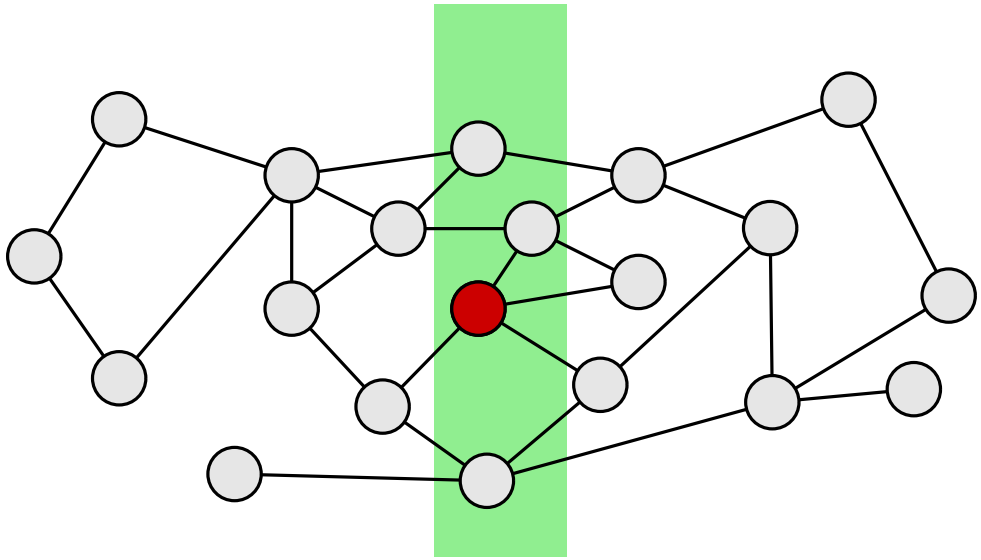
Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.



1. Compute $(2/3)$-balanced separator S of size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do
   (a) Recursively find max independent set I_A for $A \setminus \{\text{neighbors of } I_S\}$
   (b) Recursively find max independent set I_B for $B \setminus \{\text{neighbors of } I_S\}$
   (c) $I(S) := I_S \cup I_A \cup I_B$
3. Return the largest of the sets $I(S)$ found in Step 2.

Subexponential algorithms on planar graphs

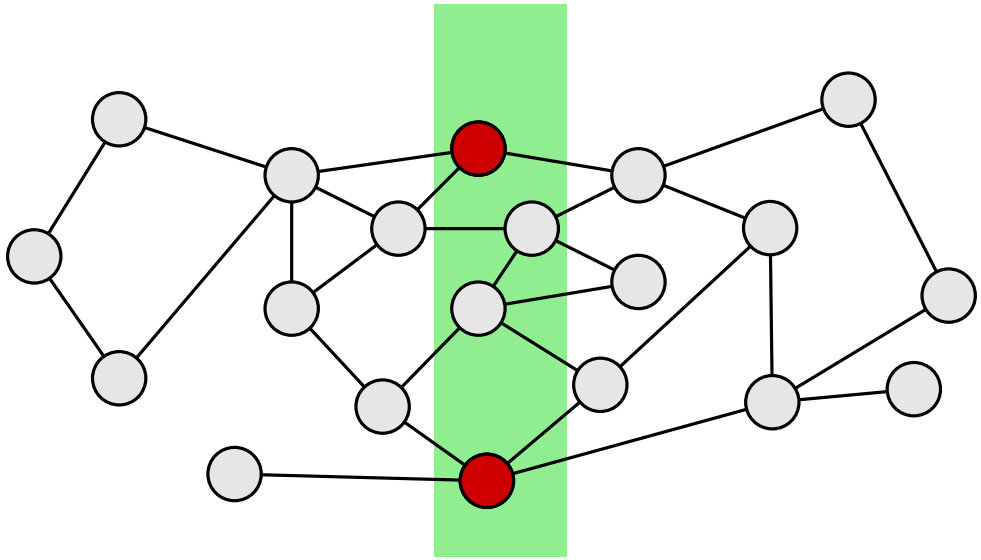
Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.



1. Compute $(2/3)$-balanced separator S of size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do
   (a) Recursively find max independent set I_A for $A \setminus \{\text{neighbors of } I_S\}$
   (b) Recursively find max independent set I_B for $B \setminus \{\text{neighbors of } I_S\}$
   (c) $I(S) := I_S \cup I_A \cup I_B$
3. Return the largest of the sets $I(S)$ found in Step 2.

Subexponential algorithms on planar graphs

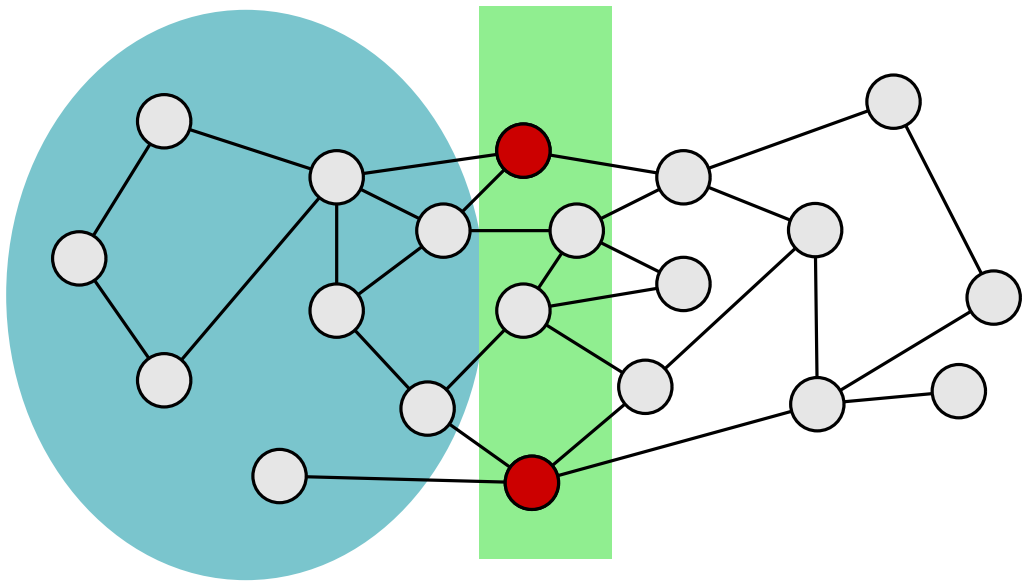
Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.



1. Compute $(2/3)$-balanced separator S of size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do
   (a) Recursively find max independent set I_A for $A \setminus \{\text{neighbors of } I_S\}$
   (b) Recursively find max independent set I_B for $B \setminus \{\text{neighbors of } I_S\}$
   (c) $I(S) := I_S \cup I_A \cup I_B$
3. Return the largest of the sets $I(S)$ found in Step 2.

Subexponential algorithms on planar graphs

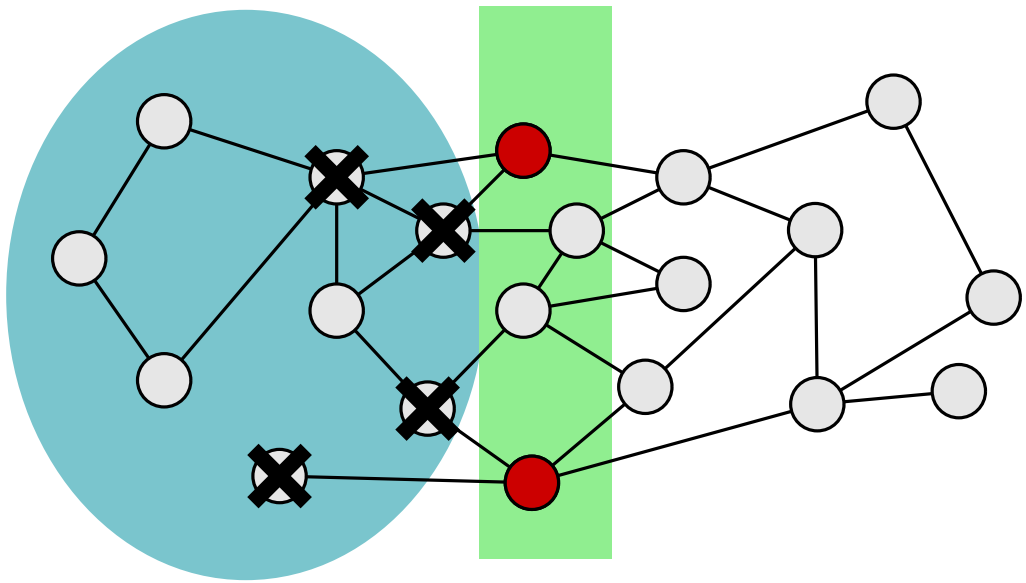
Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.



1. Compute $(2/3)$-balanced separator S of size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do
   (a) Recursively find max independent set I_A for $A \setminus \{\text{neighbors of } I_S\}$
   (b) Recursively find max independent set I_B for $B \setminus \{\text{neighbors of } I_S\}$
   (c) $I(S) := I_S \cup I_A \cup I_B$
3. Return the largest of the sets $I(S)$ found in Step 2.

Subexponential algorithms on planar graphs

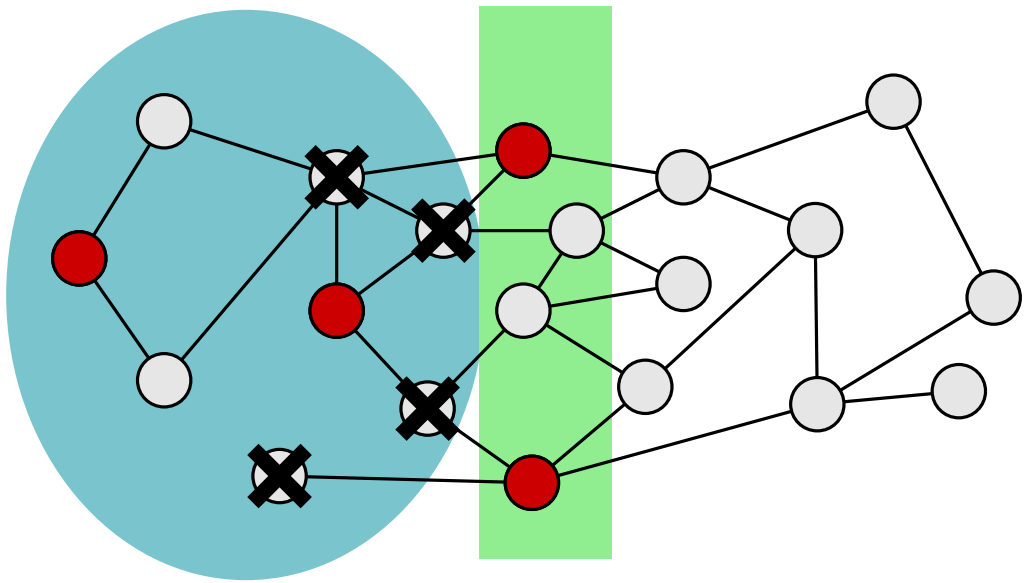
Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.



1. Compute $(2/3)$-balanced separator S of size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do
   (a) Recursively find max independent set I_A for $A \setminus \{\text{neighbors of } I_S\}$
   (b) Recursively find max independent set I_B for $B \setminus \{\text{neighbors of } I_S\}$
   (c) $I(S) := I_S \cup I_A \cup I_B$
3. Return the largest of the sets $I(S)$ found in Step 2.

Subexponential algorithms on planar graphs

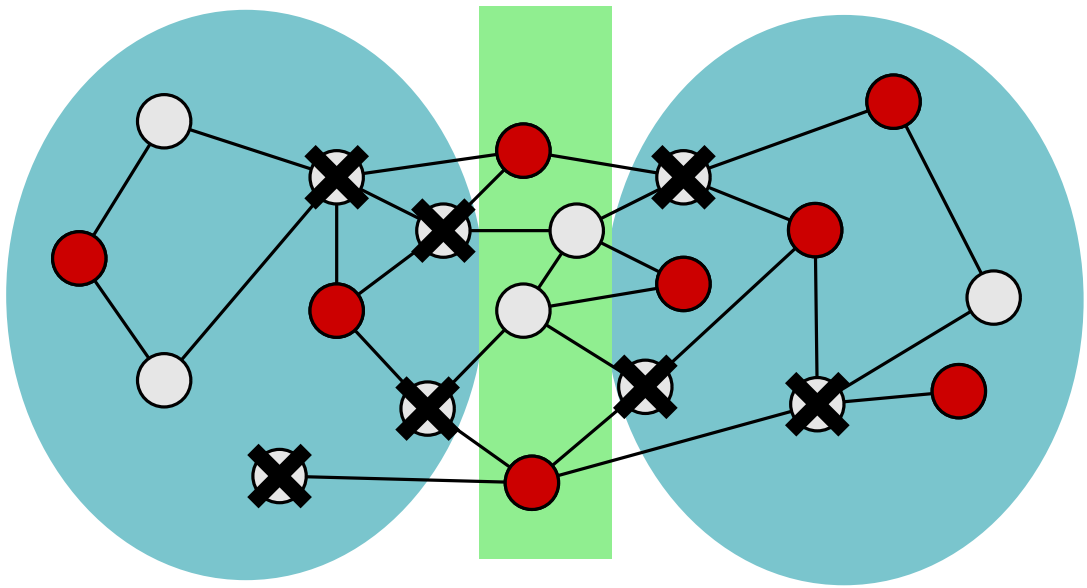
Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.



1. Compute $(2/3)$-balanced separator S of size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do
   (a) Recursively find max independent set I_A for $A \setminus \{\text{neighbors of } I_S\}$
   (b) Recursively find max independent set I_B for $B \setminus \{\text{neighbors of } I_S\}$
   (c) $I(S) := I_S \cup I_A \cup I_B$
3. Return the largest of the sets $I(S)$ found in Step 2.

Subexponential algorithms on planar graphs

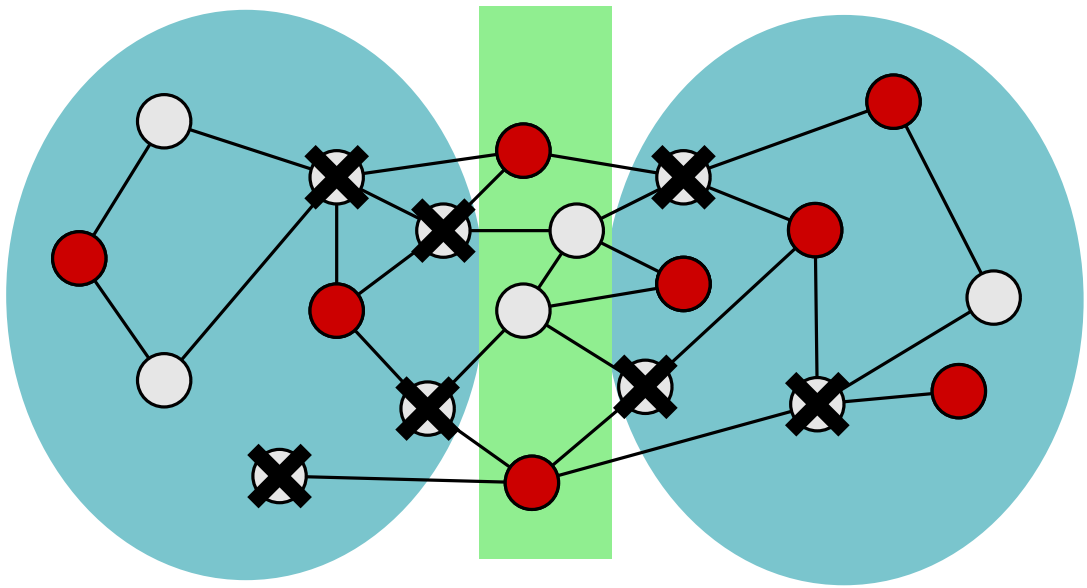
Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.



1. Compute $(2/3)$-balanced separator S of size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do
   (a) Recursively find max independent set I_A for $A \setminus \{\text{neighbors of } I_S\}$
   (b) Recursively find max independent set I_B for $B \setminus \{\text{neighbors of } I_S\}$
   (c) $I(S) := I_S \cup I_A \cup I_B$
3. Return the largest of the sets $I(S)$ found in Step 2.

Subexponential algorithms on planar graphs

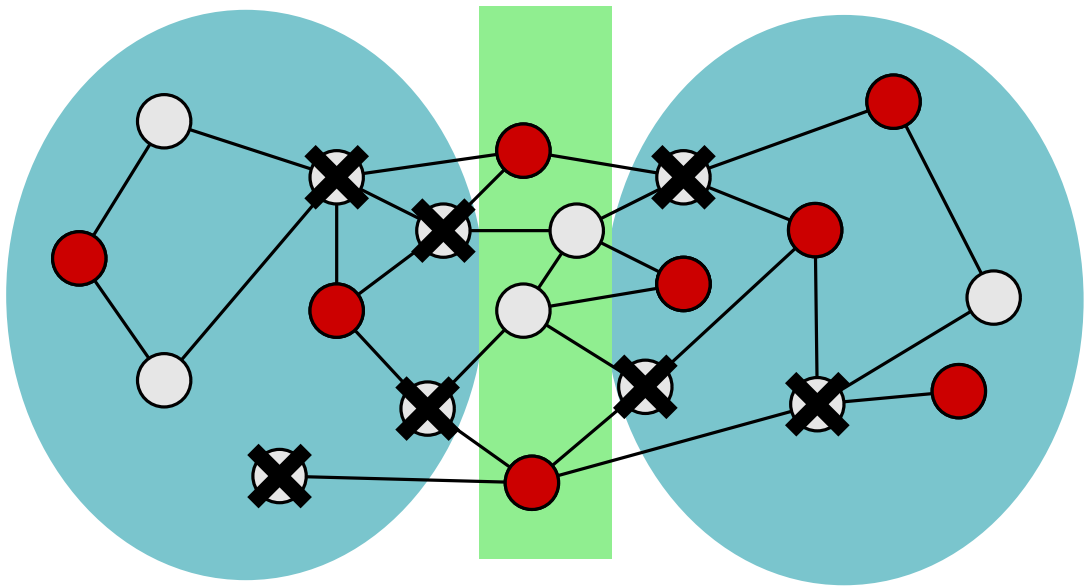
Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.



1. Compute $(2/3)$-balanced separator S of size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do
   (a) Recursively find max independent set I_A for $A \setminus \{\text{neighbors of } I_S\}$
   (b) Recursively find max independent set I_B for $B \setminus \{\text{neighbors of } I_S\}$
   (c) $I(S) := I_S \cup I_A \cup I_B$
3. Return the largest of the sets $I(S)$ found in Step 2.

Subexponential algorithms on planar graphs

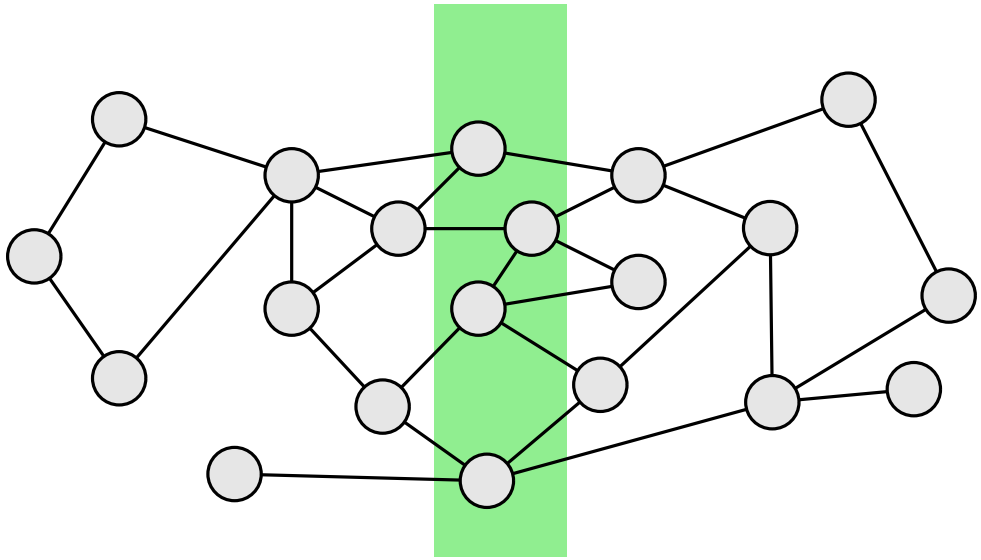
Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.



1. Compute $(2/3)$-balanced separator S of size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do
   (a) Recursively find max independent set I_A for $A \setminus \{\text{neighbors of } I_S\}$
   (b) Recursively find max independent set I_B for $B \setminus \{\text{neighbors of } I_S\}$
   (c) $I(S) := I_S \cup I_A \cup I_B$
3. Return the largest of the sets $I(S)$ found in Step 2.

Subexponential algorithms on planar graphs

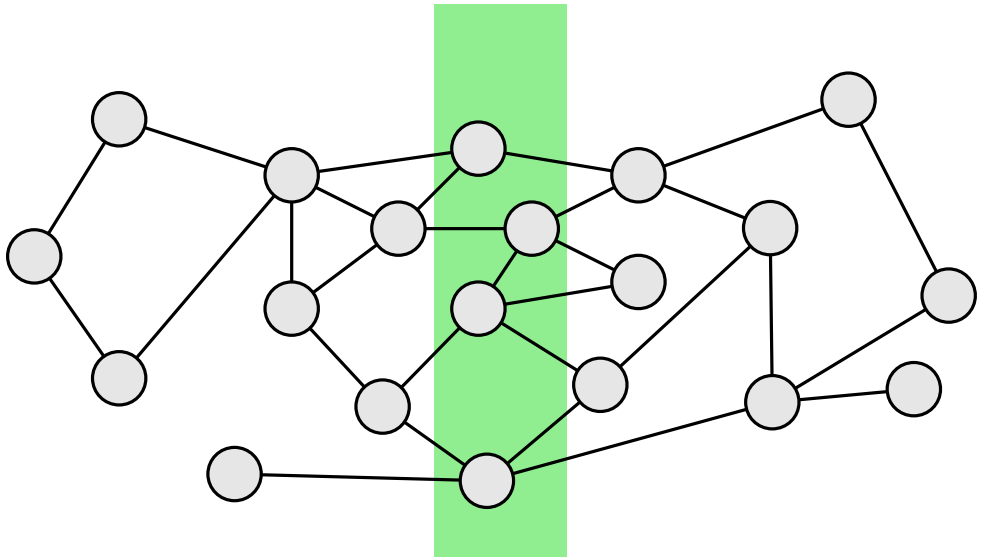
Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.



Running time

Subexponential algorithms on planar graphs

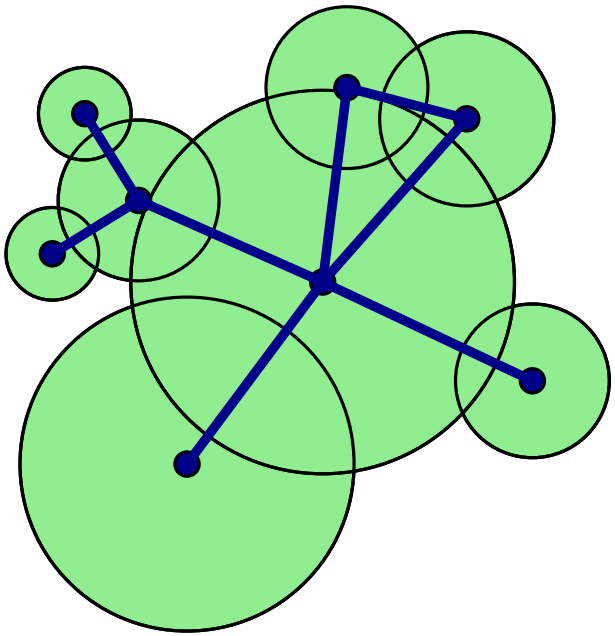
Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in planar graphs.



Running time

$$T(n) = 2^{O(\sqrt{n})} + 2^{O(\sqrt{n})} \cdot T(2n/3) \qquad \Longrightarrow \qquad T(n) = 2^{O(\sqrt{n})}$$

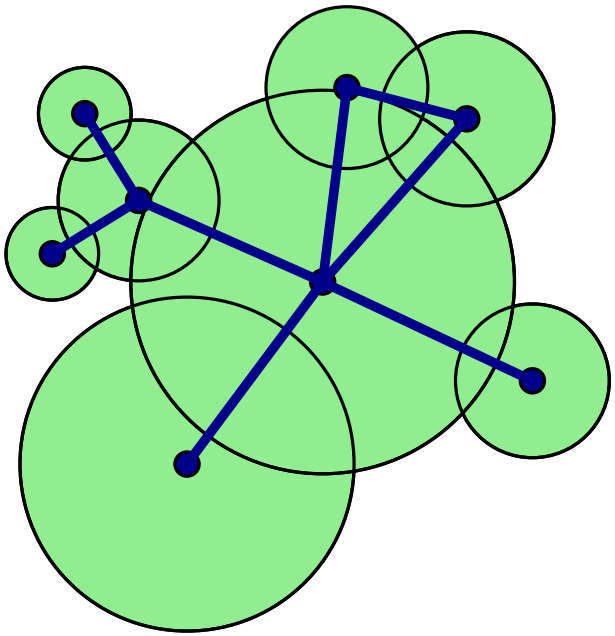
Extension to disk graphs?



Disk graph: intersection graph of set of disks:

- nodes = disks
- two nodes are connected iff the corresponding disks intersect

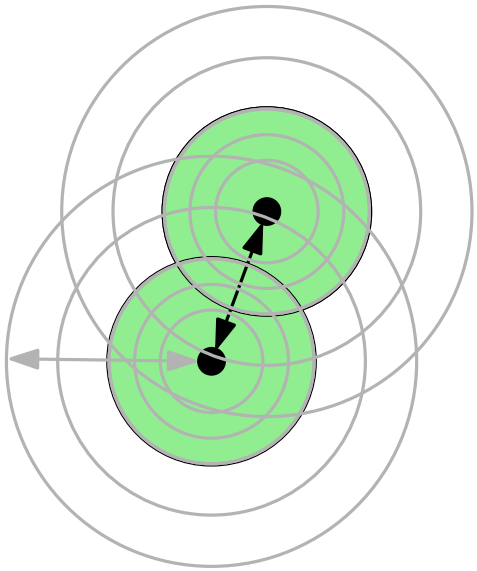
Extension to disk graphs?



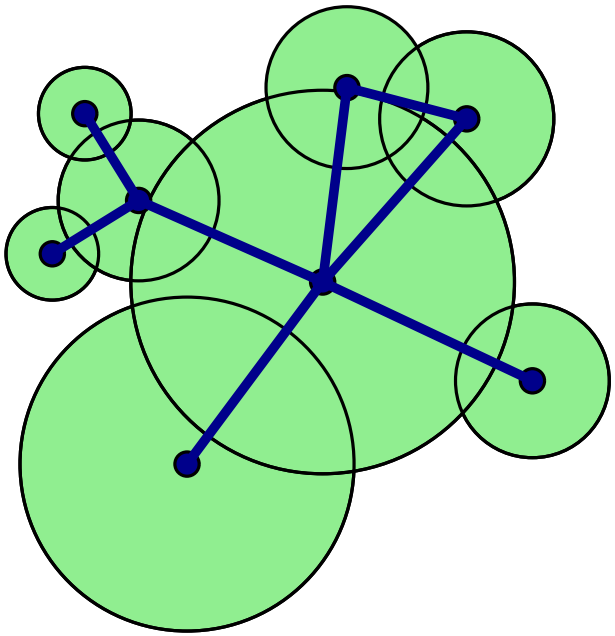
Disk graph: intersection graph of set of disks:

- nodes = disks
- two nodes are connected iff the corresponding disks intersect

Unit-disk graphs are a popular model for ad-hoc wireless networks



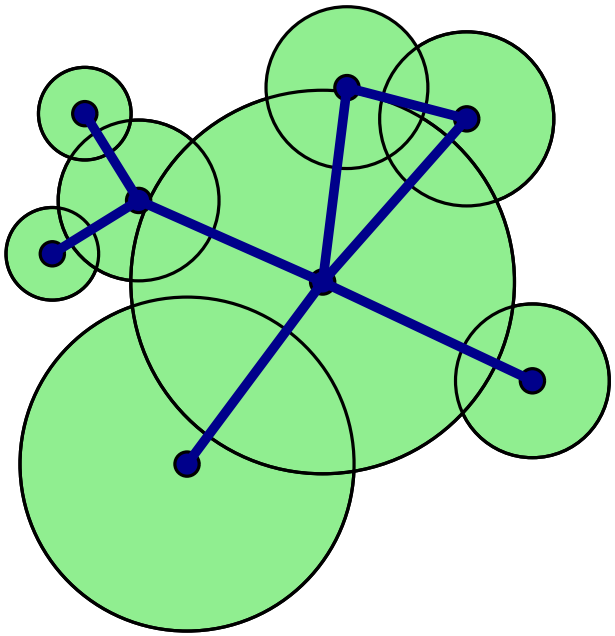
A Separator Theorem for disk graphs?



Disk graph: intersection graph of set of disks:

- nodes = disks
- two nodes are connected iff the corresponding disks intersect

A Separator Theorem for disk graphs?

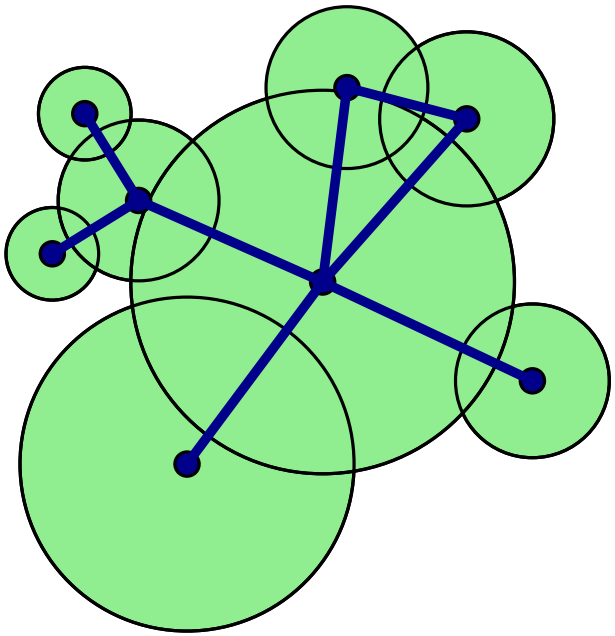


Disk graph: intersection graph of set of disks:

- nodes = disks
- two nodes are connected iff the corresponding disks intersect

Can we get a balanced separator of size $O(\sqrt{n})$ for disk graphs?

A Separator Theorem for disk graphs?

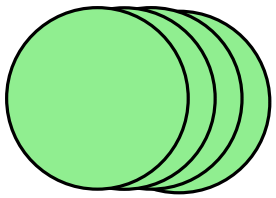


Disk graph: intersection graph of set of disks:

- nodes = disks
- two nodes are connected iff the corresponding disks intersect

Can we get a balanced separator of size $O(\sqrt{n})$ for disk graphs?

No



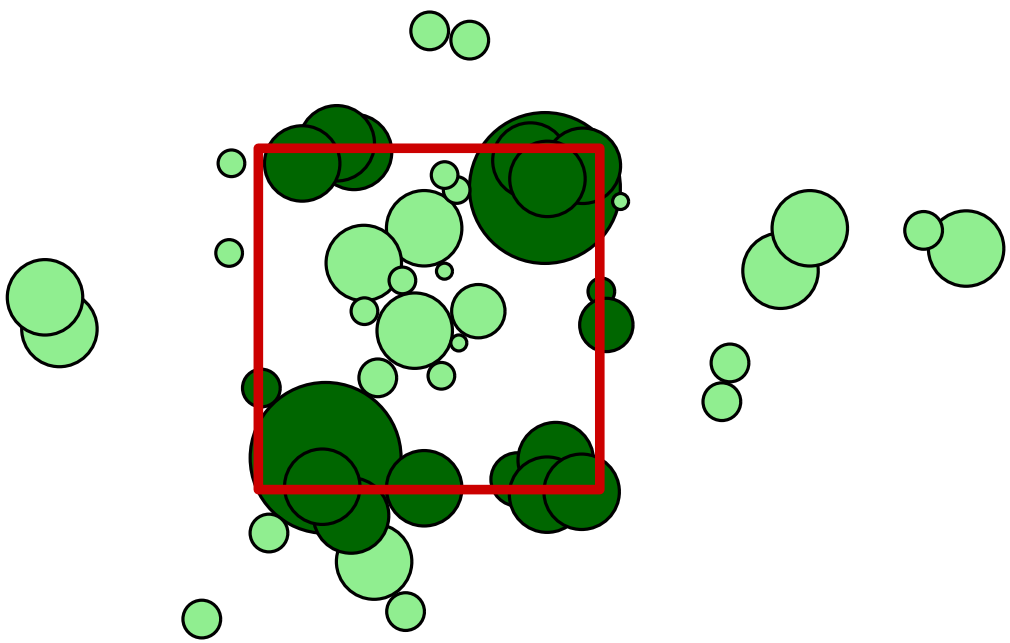
can have arbitrarily large cliques

A Separator Theorem for disk graphs

Theorem. [dB, Bodlaender, Kisfaludi-Bak, Marx, vd Zanden, STOC 2018]

Let $\mathcal{G}_D$ be the intersection graph of a set D of n disks. Then there is an α-balanced separator for $\mathcal{G}_D$, where $\alpha = 36/37$, consisting of a collection $C_1, C_2, \ldots, C_m$ of cliques such that

$$\sum_{i=1}^{m} \log(|C_i| + 1) = O(\sqrt{n})$$



Subexponential algorithms on disk graphs

Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in disk graphs.

Subexponential algorithms on disk graphs

Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in disk graphs.

1. Compute $(36/37)$-balanced separator S of weighted clique size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do

same as before

Subexponential algorithms on disk graphs

Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in disk graphs.

1. Compute $(36/37)$-balanced separator S of weighted clique size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do

same as before

Number of independent sets $I_S \subseteq S$

Subexponential algorithms on disk graphs

Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in disk graphs.

1. Compute $(36/37)$-balanced separator S of weighted clique size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do

same as before

Number of independent sets $I_S \subseteq S$

$$\Pi_{i=1}^{m}(|C_i|+1) \;=\; \Pi_{i=1}^{m} 2^{\log(|C_i|+1)} \;=\; 2^{\sum_{i=1}^{m}\log(|C_i|+1)} \;=\; 2^{O(\sqrt{n})}$$

Subexponential algorithms on disk graphs

Theorem. Independent Set can be solved in $2^{O(\sqrt{n})}$ time in disk graphs.

1. Compute $(36/37)$-balanced separator S of weighted clique size $O(\sqrt{n})$.
2. For each independent set $I_S \subseteq S$ (including empty set) do

same as before

Number of independent sets $I_S \subseteq S$

$$\Pi_{i=1}^{m}(|C_i|+1) \;=\; \Pi_{i=1}^{m} 2^{\log(|C_i|+1)} \;=\; 2^{\sum_{i=1}^{m}\log(|C_i|+1)} \;=\; 2^{O(\sqrt{n})}$$

$\Longrightarrow$ running time is $2^{O(\sqrt{n})}$

Subexponential algorithms on unit-disk graphs

Theorem. For unit-disk graphs we can also solve

- (Connected) Dominating Set
- (Connected) Vertex Cover
- (Connected) Feedback Vertex Set
- Hamiltonian Cycle

in $2^{O(\sqrt{n})}$ time.

Subexponential algorithms on unit-disk graphs

Theorem. For unit-disk graphs we can also solve

- (Connected) Dominating Set
- (Connected) Vertex Cover
- (Connected) Feedback Vertex Set
- Hamiltonian Cycle

in $2^{O(\sqrt{n})}$ time.

Approach

- use clique-based separator to get tree decomposition of small weight on intersection graph induced by cliques
- use algorithms based on (weighted) tree decompositions

All results are optimal, assuming Exponential-Time Hypothesis.

Extension to higher dimensions

Theorem. (dB, Bodlaender, Kisfaludi-Bak, Marx, van der Zanden 2018)

Let $\mathcal{G}_D$ be the intersection graph of a set D of n balls in $\mathbb{R}^d$. Then there is an α-balanced separator for $\mathcal{G}_D$, where $\alpha = 6^d/(6^d+1)$, consisting of a collection $C_1, C_2, \ldots, C_m$ of cliques such that

$$\sum_{i=1}^{m} \log(|C_i|+1) = O(n^{1-1/d})$$

Extension to higher dimensions

Theorem. (dB, Bodlaender, Kisfaludi-Bak, Marx, van der Zanden 2018)

Let $\mathcal{G}_D$ be the intersection graph of a set D of n balls in $\mathbb{R}^d$. Then there is an α-balanced separator for $\mathcal{G}_D$, where $\alpha = 6^d/(6^d+1)$, consisting of a collection $C_1, C_2, \ldots, C_m$ of cliques such that

$$\sum_{i=1}^{m} \log(|C_i|+1) = O(n^{1-1/d})$$

Why $n^{1-1/d}$?

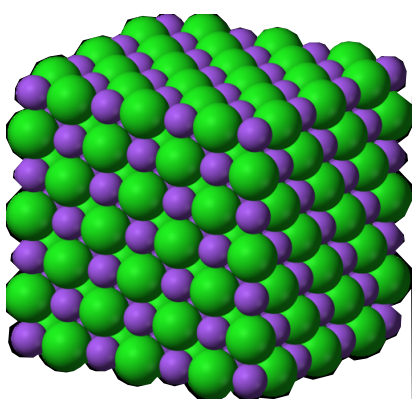
Extension to higher dimensions

Theorem. (dB, Bodlaender, Kisfaludi-Bak, Marx, van der Zanden 2018)

Let $\mathcal{G}_D$ be the intersection graph of a set D of n balls in $\mathbb{R}^d$. Then there is an α-balanced separator for $\mathcal{G}_D$, where $\alpha = 6^d/(6^d+1)$, consisting of a collection $C_1, C_2, \ldots, C_m$ of cliques such that

$$\sum_{i=1}^{m} \log(|C_i|+1) = O(n^{1-1/d})$$

Why $n^{1-1/d}$?



From: chemistry.stackexchange.com

- $n^{1/d} \times \cdots \times n^{1/d}$ spheres
- any balanced separator has size $\Omega(n^{1-1/d})$

Extension to higher dimensions

Theorem.

For ball graphs we can solve Independent Set in $2^{O(n^{1-1/d})}$ time.

For unit-ball graphs we can also solve

- (Connected) Dominating Set
- (Connected) Vertex Cover
- (Connected) Feedback Vertex Set
- Hamiltonian Cycle

in $2^{O(n^{1-1/d})}$ time.

Also works for (similarly-sized) fat objects.

Extension to higher dimensions

Theorem.

For ball graphs we can solve Independent Set in $2^{O(n^{1-1/d})}$ time.

For unit-ball graphs we can also solve

- (Connected) Dominating Set
- (Connected) Vertex Cover
- (Connected) Feedback Vertex Set
- Hamiltonian Cycle

in $2^{O(n^{1-1/d})}$ time.

Also works for (similarly-sized) fat objects.

All these results are optimal, assuming ETH.

End of commercial break

Stable Approximation Schemes for **Independent Set**

TU/e

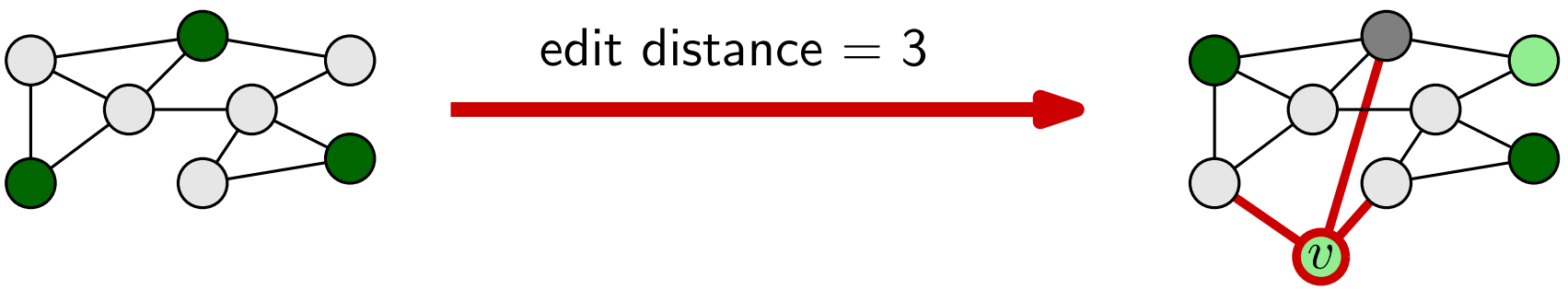
NET WORKS

Stable Algorithms for Independent Set

joint work with Arpan Sadhukan and Frits Speiksma (in preparation)

The model

- vertex arrival model: vertex is inserted or deleted with all its incident edges
- edit distance: number of additions plus removals in independent set



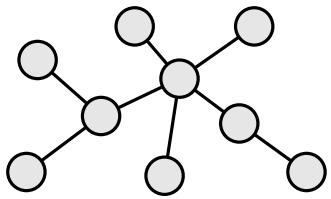
solution before
arrival of vertex v

solution after
arrival of vertex v

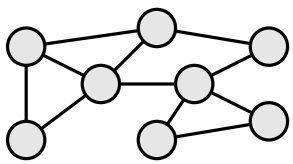
Stable Approximation Schemes for Independent Set?

Is there a SAS for Independent Set on

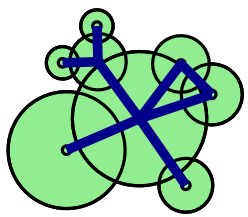
- trees?



- graphs of maximum bounded degree?
- planar graphs?

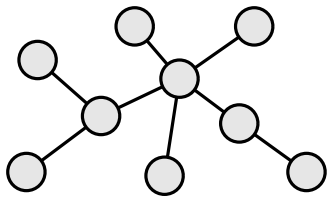
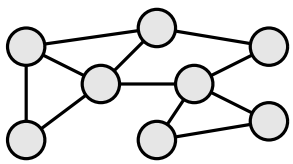
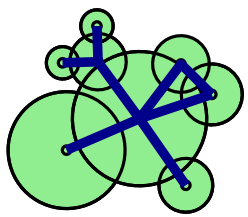


- disk graphs?



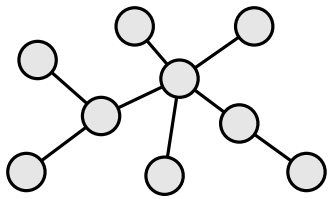
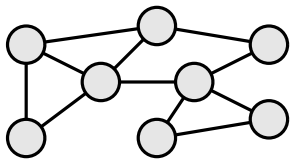
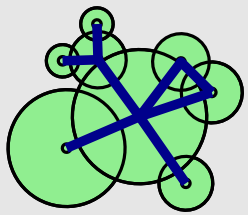
Stable Approximation Schemes for Independent Set?

Is there a SAS for Independent Set on

- trees?  **YES**
- graphs of maximum bounded degree? **NO**
- planar graphs?  **YES**
- disk graphs?  **YES**

Stable Approximation Schemes for Independent Set?

Is there a SAS for Independent Set on

- trees?  **YES**
- graphs of maximum bounded degree? **NO** 
- planar graphs?  **YES**
- disk graphs?  **YES**

No SAS for bounded-degree graphs

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

No SAS for bounded-degree graphs

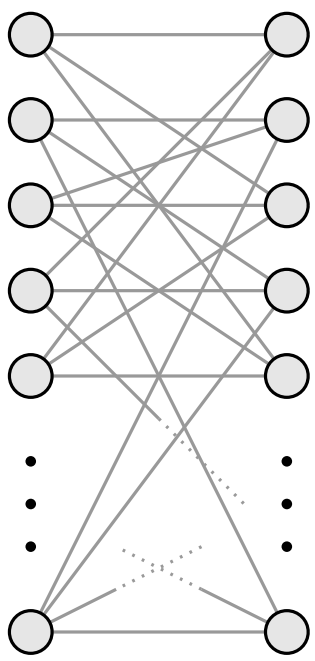
Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

Proof. There is an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation requires $k_\varepsilon = \Omega(n)$

No SAS for bounded-degree graphs

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

Proof. There is an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation requires $k_\varepsilon = \Omega(n)$



$|L| = n \quad |R| = n$

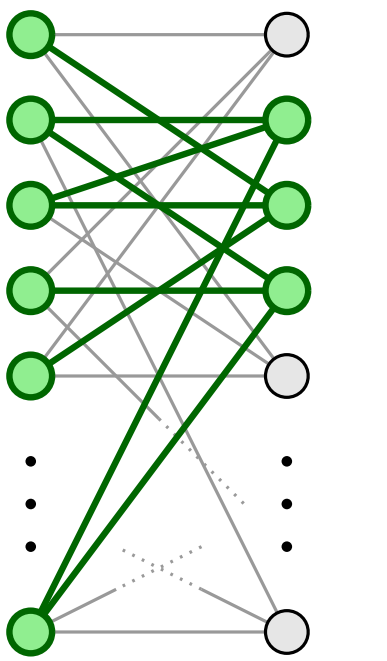
3-regular bipartite expander with $2n$ vertices

- every node has degree 3
- $|N[S]| \geqslant 1.99|S|$ for all $S \subset V$ of size at most cn for some $c > 0$

No SAS for bounded-degree graphs

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

Proof. There is an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation requires $k_\varepsilon = \Omega(n)$



$|L| = n \quad |R| = n$

3-regular bipartite expander with $2n$ vertices

- every node has degree 3
- $|N[S]| \geqslant 1.99|S|$ for all $S \subset V$ of size at most cn for some $c > 0$

No SAS for bounded-degree graphs

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

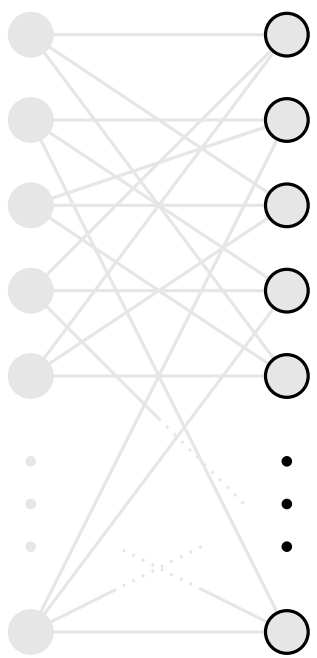
Proof. There is an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation requires $k_\varepsilon = \Omega(n)$

$|L| = n \quad |R| = n$

No SAS for bounded-degree graphs

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

Proof. There is an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation requires $k_\varepsilon = \Omega(n)$



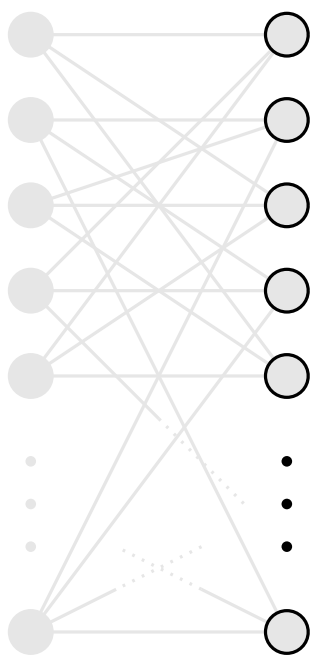
$|L| = n \quad |R| = n$

- first R arrives

No SAS for bounded-degree graphs

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

Proof. There is an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation requires $k_\varepsilon = \Omega(n)$



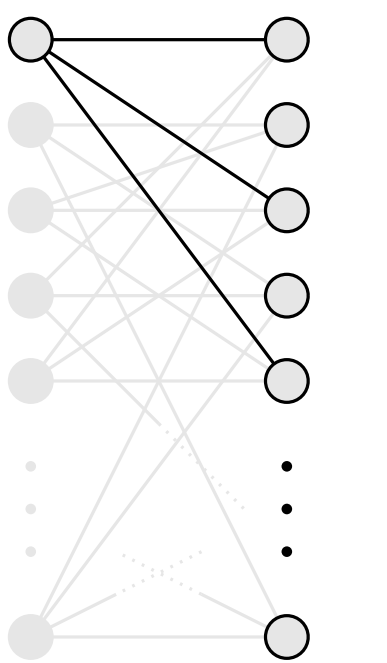
$|L| = n \quad |R| = n$

- first R arrives
- then L arrives

No SAS for bounded-degree graphs

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

Proof. There is an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation requires $k_\varepsilon = \Omega(n)$



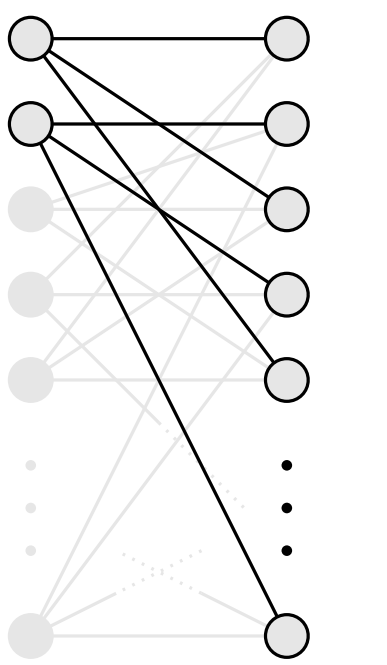
$|L| = n \quad |R| = n$

- first R arrives
- then L arrives

No SAS for bounded-degree graphs

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

Proof. There is an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation requires $k_\varepsilon = \Omega(n)$



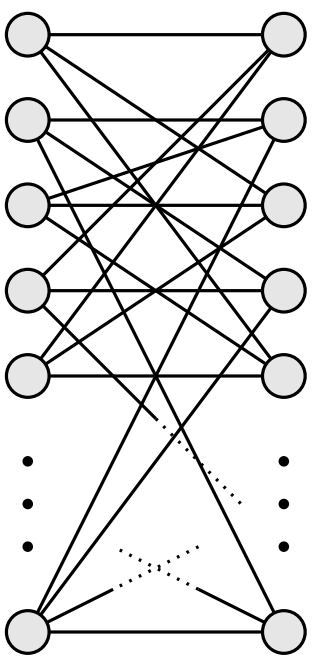
$|L| = n \quad |R| = n$

- first R arrives
- then L arrives

No SAS for bounded-degree graphs

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

Proof. There is an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation requires $k_\varepsilon = \Omega(n)$



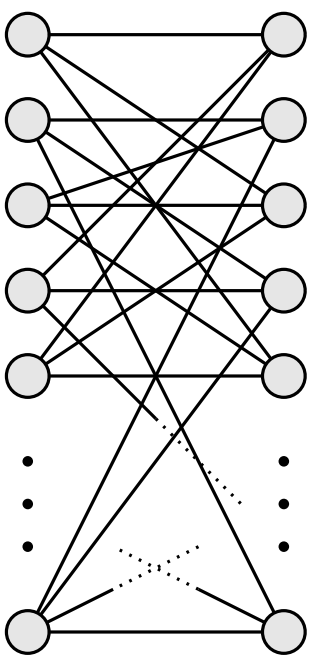
$|L| = n \quad |R| = n$

- first R arrives
- then L arrives

No SAS for bounded-degree graphs

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

Proof. There is an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation requires $k_\varepsilon = \Omega(n)$



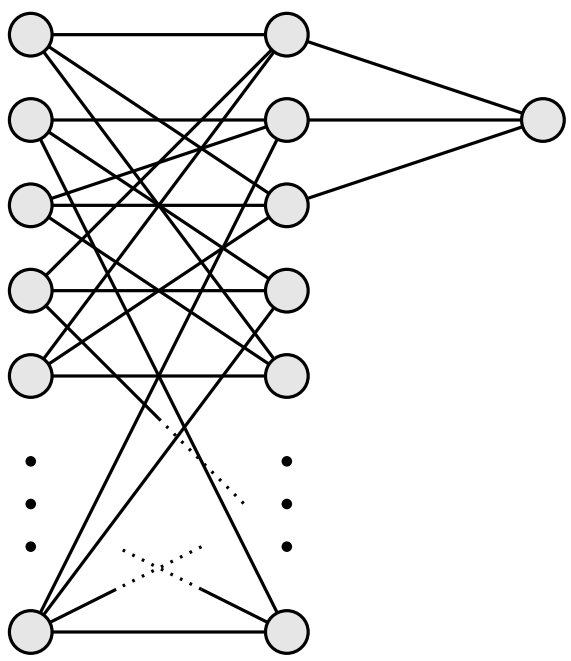
$|L| = n \quad |R| = n$

- first R arrives
- then L arrives
- then $n/3$ degree-3 vertices arrive, connecting to distinct vertices in R

No SAS for bounded-degree graphs

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

Proof. There is an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation requires $k_\varepsilon = \Omega(n)$



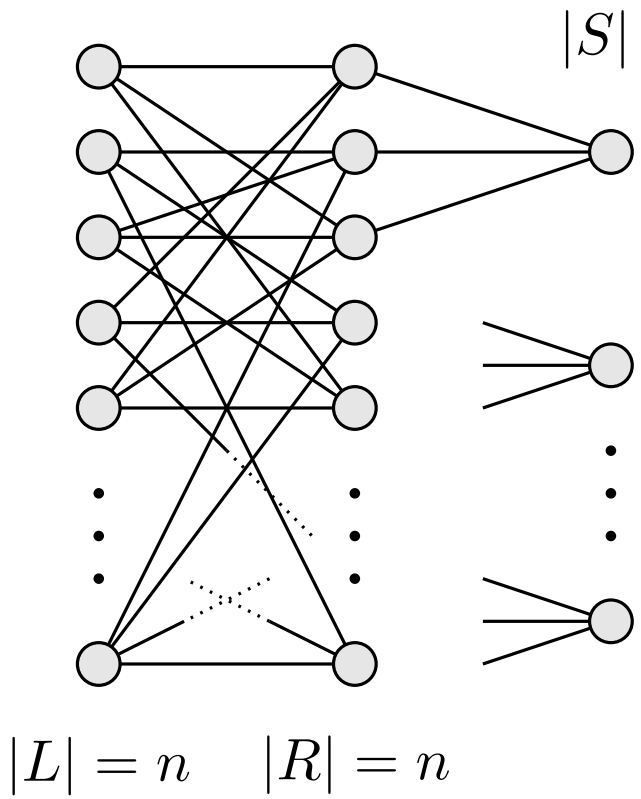
$|L| = n \quad |R| = n$

- first R arrives
- then L arrives
- then $n/3$ degree-3 vertices arrive, connecting to distinct vertices in R

No SAS for bounded-degree graphs

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

Proof. There is an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation requires $k_\varepsilon = \Omega(n)$

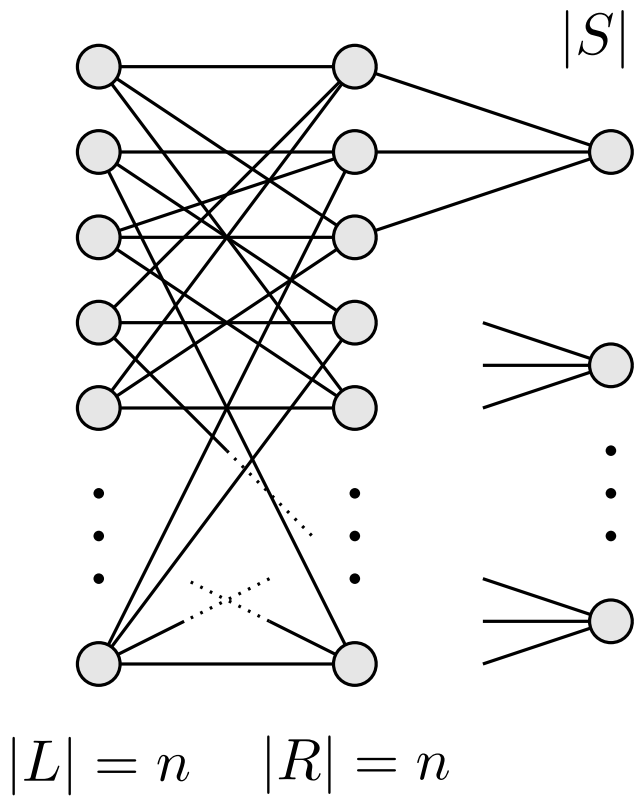


- first R arrives
- then L arrives
- then $n/3$ degree-3 vertices arrive, connecting to distinct vertices in R

No SAS for bounded-degree graphs

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

Proof. There is an $\varepsilon > 0$ such that $(1-\varepsilon)$-approximation requires $k_\varepsilon = \Omega(n)$



- first R arrives
- then L arrives
- then $n/3$ degree-3 vertices arrive, connecting to distinct vertices in R

Claim. When $k_\varepsilon = o(n)$ then independent set I_{alg} will always contain less than $cn/16$ vertices from $L \cup S$.

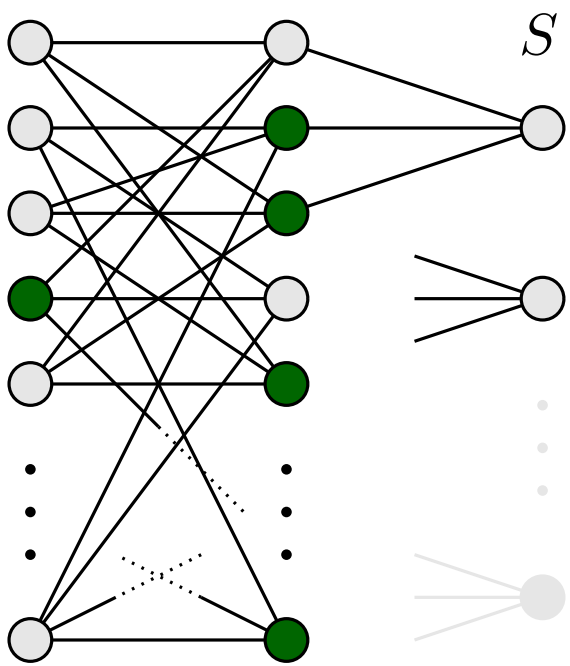
No SAS for bounded-degree graphs

Claim. When $k_\varepsilon = o(n)$ then independent set I_{alg} will always contain less than $cn/16$ vertices from $L \cup S$.

No SAS for bounded-degree graphs

Claim. When $k_\varepsilon = o(n)$ then independent set I_{alg} will always contain less than $cn/16$ vertices from $L \cup S$.

Proof by contradition.

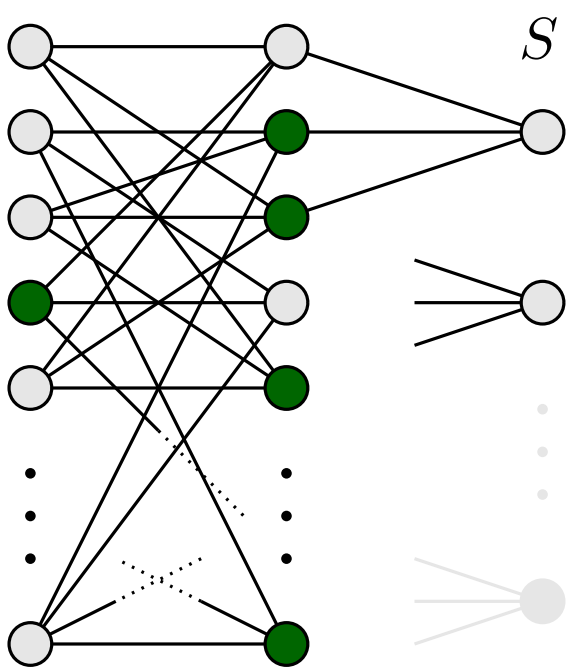


Consider first time $|I_{\text{alg}} \cap (L \cup S)| \geqslant cn/16$

No SAS for bounded-degree graphs

Claim. When $k_\varepsilon = o(n)$ then independent set I_{alg} will always contain less than $cn/16$ vertices from $L \cup S$.

Proof by contradition.



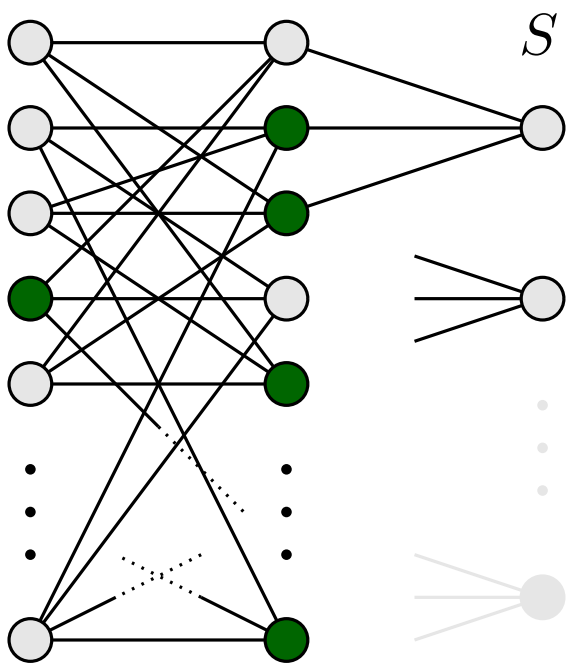
Consider first time $|I_{\text{alg}} \cap (L \cup S)| \geqslant cn/16$

$$|I_{\text{alg}} \cap (L \cup S)| \leqslant cn/16 + k_\varepsilon \approx cn/16$$

No SAS for bounded-degree graphs

Claim. When $k_\varepsilon = o(n)$ then independent set I_{alg} will always contain less than $cn/16$ vertices from $L \cup S$.

Proof by contradition.



Consider first time $|I_{\text{alg}} \cap (L \cup S)| \geqslant cn/16$

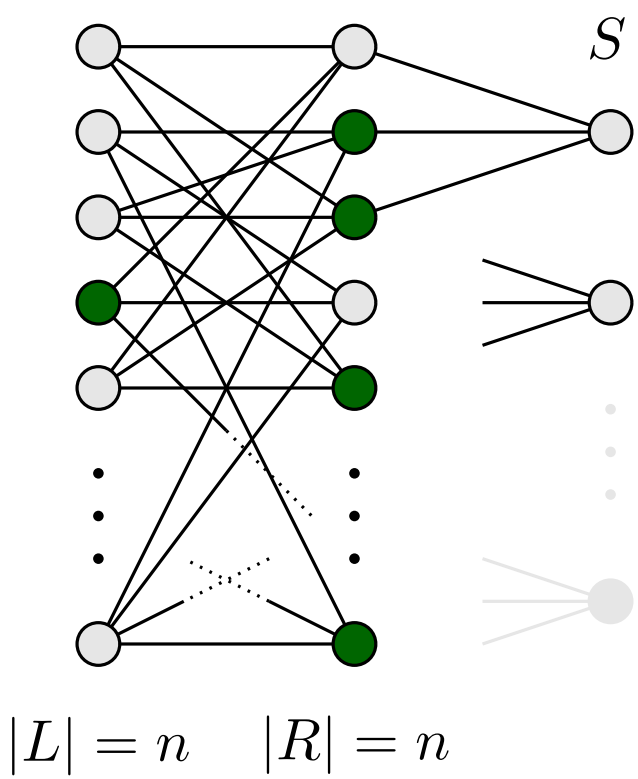
$$|I_{\text{alg}} \cap (L \cup S)| \leqslant cn/16 + k_\varepsilon \approx cn/16$$

Case (i): $|I_{\text{alg}} \cap L| \geqslant \frac{7}{12} \cdot cn/16$

No SAS for bounded-degree graphs

Claim. When $k_\varepsilon = o(n)$ then independent set I_{alg} will always contain less than $cn/16$ vertices from $L \cup S$.

Proof by contradition.



Consider first time $|I_{\text{alg}} \cap (L \cup S)| \geqslant cn/16$

$$|I_{\text{alg}} \cap (L \cup S)| \leqslant cn/16 + k_\varepsilon \approx cn/16$$

Case (i): $|I_{\text{alg}} \cap L| \geqslant \frac{7}{12} \cdot cn/16$

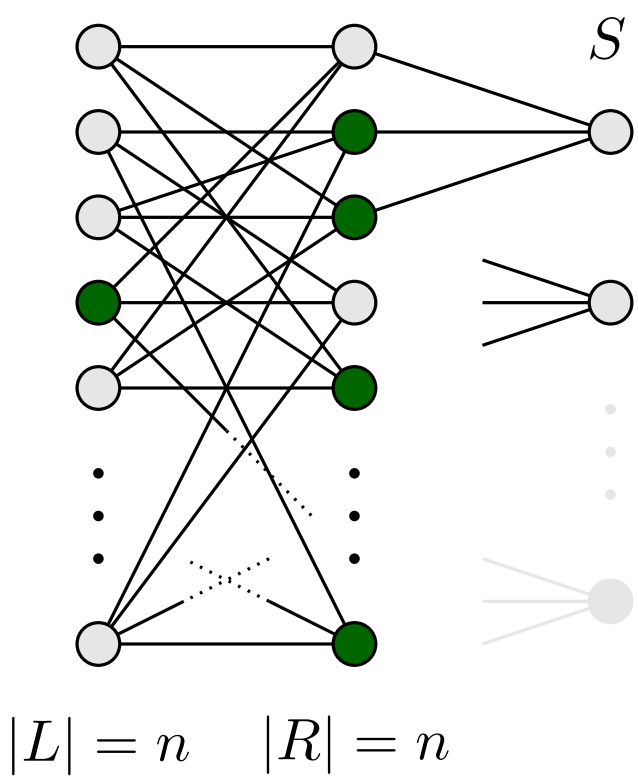
Then

$$\begin{aligned} |I_{\text{alg}}| &= |I_{\text{alg}} \cap (L \cup S)| + |I_{\text{alg}} \cap R| \\ &\leqslant cn/16 + (n - 1.99 \cdot \tfrac{7}{12} \cdot cn/16) \\ &\leqslant n - \tfrac{1}{12} \cdot cn/16 \end{aligned}$$

No SAS for bounded-degree graphs

Claim. When $k_\varepsilon = o(n)$ then independent set I_{alg} will always contain less than $cn/16$ vertices from $L \cup S$.

Proof by contradition.



Consider first time $|I_{\text{alg}} \cap (L \cup S)| \geqslant cn/16$

$$|I_{\text{alg}} \cap (L \cup S)| \leqslant cn/16 + k_\varepsilon \approx cn/16$$

Case (i): $|I_{\text{alg}} \cap L| \geqslant \frac{7}{12} \cdot cn/16$

Then

$$\begin{aligned} |I_{\text{alg}}| &= |I_{\text{alg}} \cap (L \cup S)| + |I_{\text{alg}} \cap R| \\ &\leqslant cn/16 + (n - 1.99 \cdot \tfrac{7}{12} \cdot cn/16) \\ &\leqslant n - \tfrac{1}{12} \cdot cn/16 \end{aligned}$$

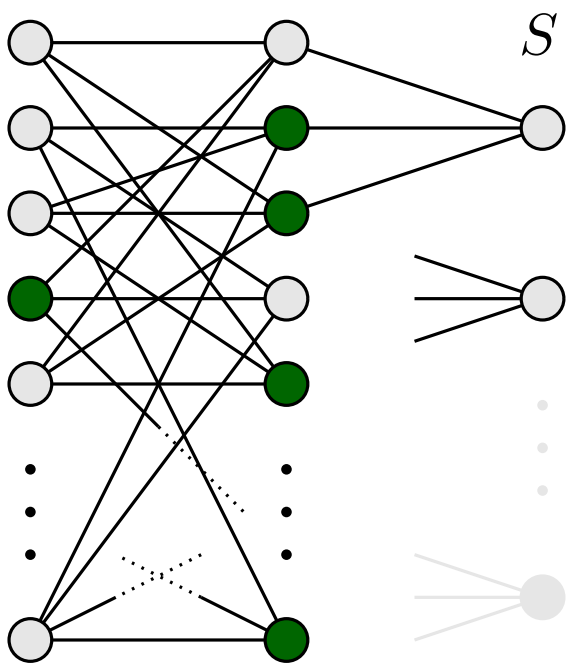
$\textsc{Opt} \geqslant n \Longrightarrow$ approximation ratio not $(1 - \varepsilon)$

for sufficiently small ε

No SAS for bounded-degree graphs

Claim. When $k_\varepsilon = o(n)$ then independent set I_{alg} will always contain less than $cn/16$ vertices from $L \cup S$.

Proof by contradition.



Consider first time $|I_{\text{alg}} \cap (L \cup S)| \geqslant cn/16$

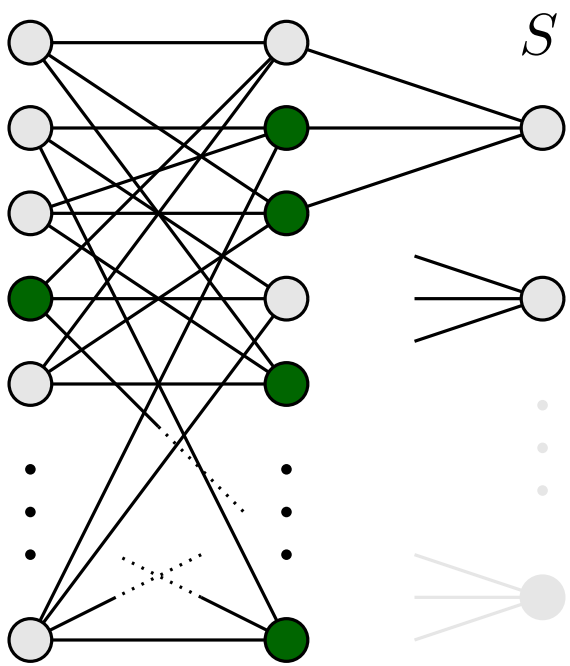
$$|I_{\text{alg}} \cap (L \cup S)| \leqslant cn/16 + k_\varepsilon \approx cn/16$$

Case (ii): $|I_{\text{alg}} \cap L| < \frac{7}{12} \cdot cn/16$

No SAS for bounded-degree graphs

Claim. When $k_\varepsilon = o(n)$ then independent set I_{alg} will always contain less than $cn/16$ vertices from $L \cup S$.

Proof by contradition.



Consider first time $|I_{\text{alg}} \cap (L \cup S)| \geqslant cn/16$

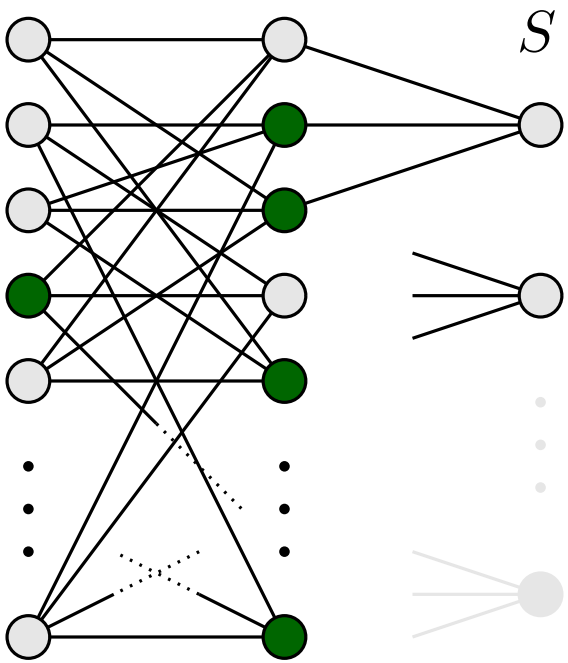
$$|I_{\text{alg}} \cap (L \cup S)| \leqslant cn/16 + k_\varepsilon \approx cn/16$$

Case (ii): $|I_{\text{alg}} \cap L| < \frac{7}{12} \cdot cn/16$

similar

No SAS for bounded-degree graphs

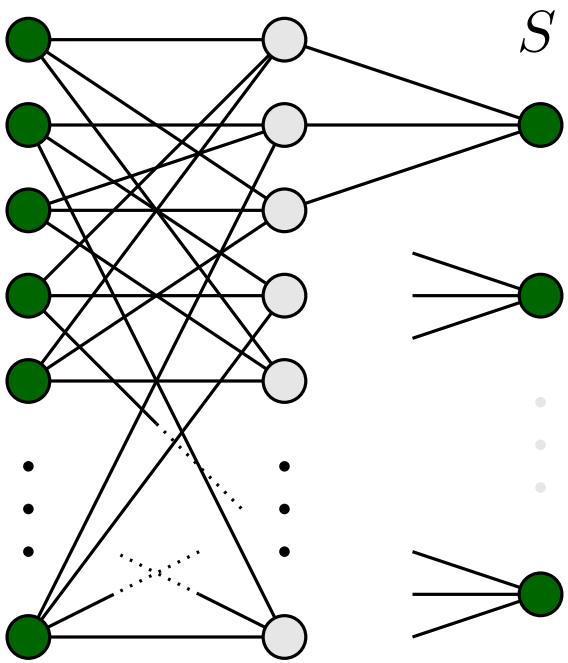
Claim. When $k_\varepsilon = o(n)$ then independent set I_{alg} will always contain less than $cn/16$ vertices from $L \cup S$.



$|L| = n \quad |R| = n$

No SAS for bounded-degree graphs

Claim. When $k_\varepsilon = o(n)$ then independent set I_{alg} will always contain less than $cn/16$ vertices from $L \cup S$.



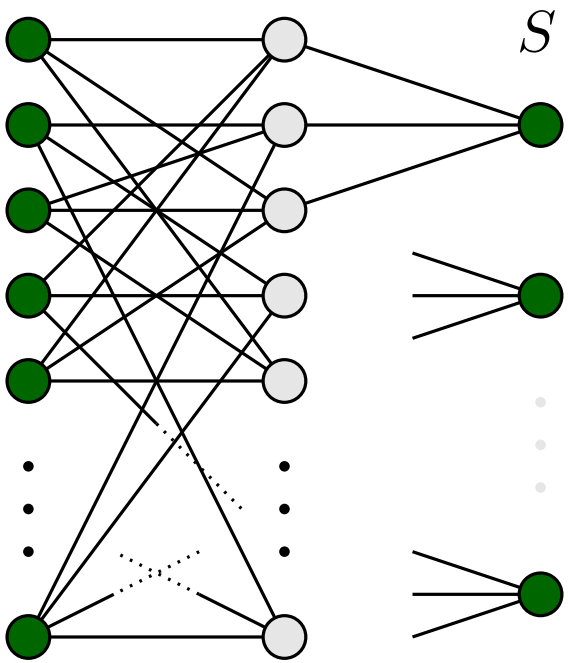
$|L| = n \quad |R| = n$

at the end: $\text{OPT} = \frac{4}{3} \cdot n$

and $|I_{\text{alg}}| < \left(1 + \frac{c}{16}\right) n$ where $0 < c < 1$

No SAS for bounded-degree graphs

Claim. When $k_\varepsilon = o(n)$ then independent set I_{alg} will always contain less than $cn/16$ vertices from $L \cup S$.



$|L| = n \quad |R| = n$

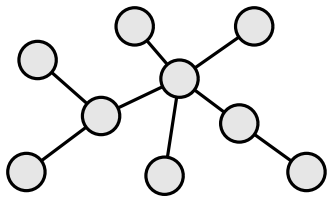
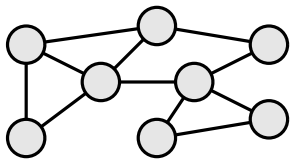
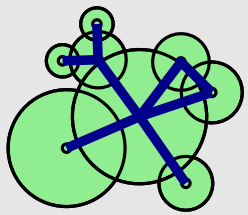
at the end: $\textsc{Opt} = \frac{4}{3} \cdot n$

and $|I_{\text{alg}}| < \left(1 + \frac{c}{16}\right) n$ where $0 < c < 1$

Theorem. The class of bipartite graphs of maximum degree 4 does not admit a SAS for Independent Set.

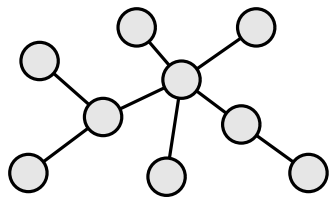
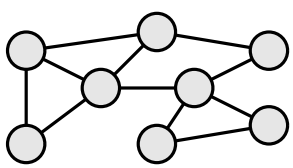
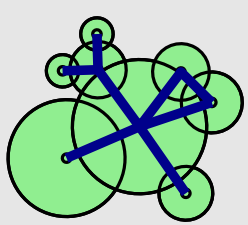
Stable Approximation Schemes for Independent Set?

Is there a SAS for Independent Set on

- trees?  **YES**
- graphs of maximum bounded degree? **NO** 
- planar graphs?  **YES**
- disk graphs?  **YES**

Stable Approximation Schemes for Independent Set?

Is there a SAS for Independent Set on

- trees?  **YES**
- graphs of maximum bounded degree? **NO**
- planar graphs?  **YES**
- disk graphs?  **YES** 

SAS for graphs with small clique-based separators

Theorem. There is a SAS for disk graphs (and other graphs with sublinear clique-based separators) for Independent Set.

SAS for graphs with small clique-based separators

Theorem. There is a SAS for disk graphs (and other graphs with sublinear clique-based separators) for Independent Set.

Consider graph class with clique-based separators of weight $O(n^\delta)$, for $\delta < 1$.

Algorithm

▷ Upon each insertion or deletion, do the following

1. **if** $|I_{\text{alg}}| \geqslant (1-\varepsilon) \cdot \text{Opt}$
2. **then** do nothing
3. **else** find $J_{\text{out}}, J_{\text{in}}$ such that
   - $(I_{\text{alg}} \setminus J_{\text{out}}) \cup J_{\text{in}}$ is independent set
   - $|J_{\text{in}}| > |J_{\text{out}}|$
   - $|J_{\text{out}}| + |J_{\text{in}}| = O((1/\varepsilon)^{\frac{1}{1-\delta}})$
4. replace J_{out} by J_{in}

SAS for graphs with small clique-based separators

Theorem. There is a SAS for disk graphs (and other graphs with sublinear clique-based separators) for Independent Set.

Consider graph class with clique-based separators of weight $O(n^\delta)$, for $\delta < 1$.

Algorithm

▷ Upon each insertion or deletion, do the following

1. **if** $|I_{\text{alg}}| \geqslant (1-\varepsilon) \cdot \text{Opt}$
2. **then** do nothing
3. **else** find $J_{\text{out}}, J_{\text{in}}$ such that
   - $(I_{\text{alg}} \setminus J_{\text{out}}) \cup J_{\text{in}}$ is independent set
   - $|J_{\text{in}}| > |J_{\text{out}}|$
   - $|J_{\text{out}}| + |J_{\text{in}}| = O((1/\varepsilon)^{\frac{1}{1-\delta}})$
4. replace J_{out} by J_{in} $\implies$ will restore $|I_{\text{alg}}| \geqslant (1-\varepsilon) \cdot \text{Opt}$

SAS for graphs with small clique-based separators

Theorem. There is a SAS for disk graphs (and other graphs with sublinear clique-based separators) for Independent Set.

Consider graph class with clique-based separators of weight $O(n^\delta)$, for $\delta < 1$.

Algorithm

▷ Upon each insertion or deletion, do the following

1. **if** $|I_{\text{alg}}| \geqslant (1-\varepsilon) \cdot \text{Opt}$
2. **then** do nothing
3. **else** find $J_{\text{out}}, J_{\text{in}}$ such that
   - $(I_{\text{alg}} \setminus J_{\text{out}}) \cup J_{\text{in}}$ is independent set
   - $|J_{\text{in}}| > |J_{\text{out}}|$
   - $|J_{\text{out}}| + |J_{\text{in}}| = O((1/\varepsilon)^{\frac{1}{1-\delta}})$

   always possible?
4. replace J_{out} by J_{in} $\implies$ will restore $|I_{\text{alg}}| \geqslant (1-\varepsilon) \cdot \text{Opt}$

SAS for graphs with small clique-based separators

Lemma. Let I_{alg} be an independent set such that $|I_{\text{alg}}| < (1-\varepsilon) \cdot \textsc{Opt}$. Then there are $J_{\text{out}} \subseteq I_{\text{alg}}$ and J_{in} such that

- $(I_{\text{alg}} \setminus J_{\text{out}}) \cup J_{\text{in}}$ is independent set
- $|J_{\text{in}}| > |J_{\text{out}}|$
- $|J_{\text{out}}| + |J_{\text{in}}| = O((1/\varepsilon)^{\frac{1}{1-\delta}})$

SAS for graphs with small clique-based separators

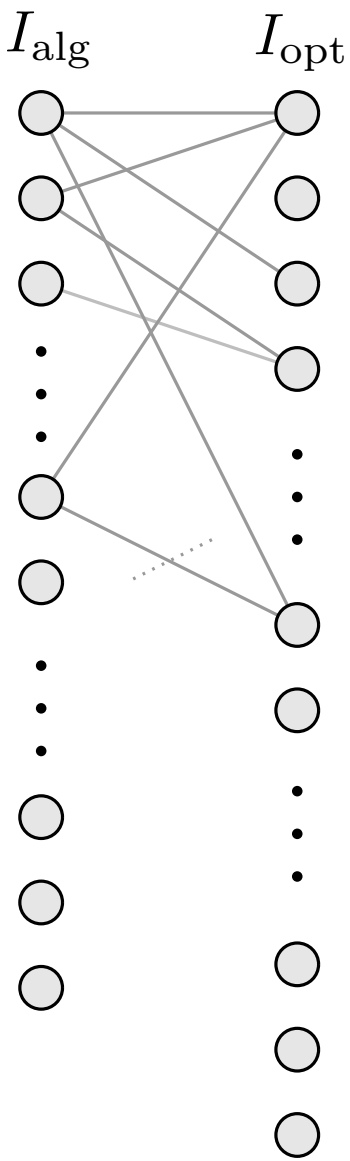
Lemma. Let I_{alg} be an independent set such that $|I_{\text{alg}}| < (1-\varepsilon) \cdot \textsc{Opt}$. Then there are $J_{\text{out}} \subseteq I_{\text{alg}}$ and J_{in} such that

- $(I_{\text{alg}} \setminus J_{\text{out}}) \cup J_{\text{in}}$ is independent set
- $|J_{\text{in}}| > |J_{\text{out}}|$
- $|J_{\text{out}}| + |J_{\text{in}}| = O((1/\varepsilon)^{\frac{1}{1-\delta}})$

- also basis for local-search PTAS
- proof using clique-based separators (similar to some existing proofs)

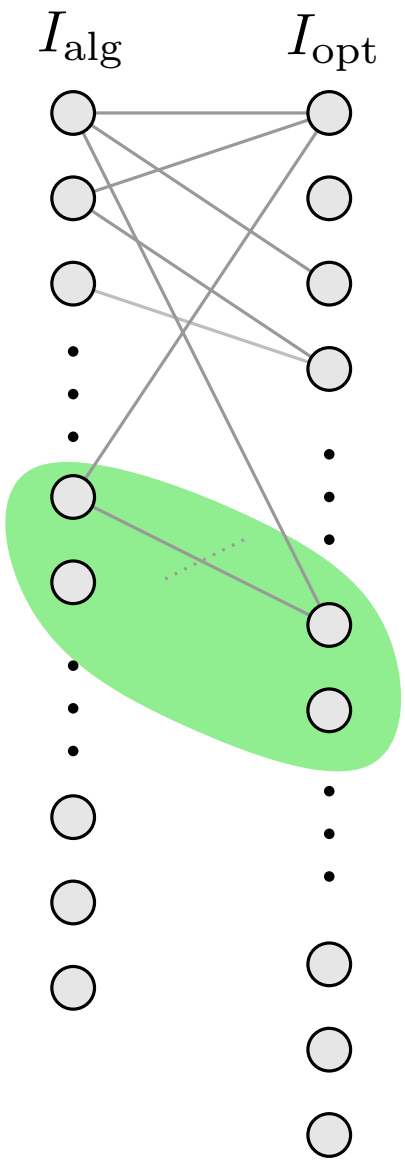
SAS for graphs with small clique-based separators

Proof sketch.



SAS for graphs with small clique-based separators

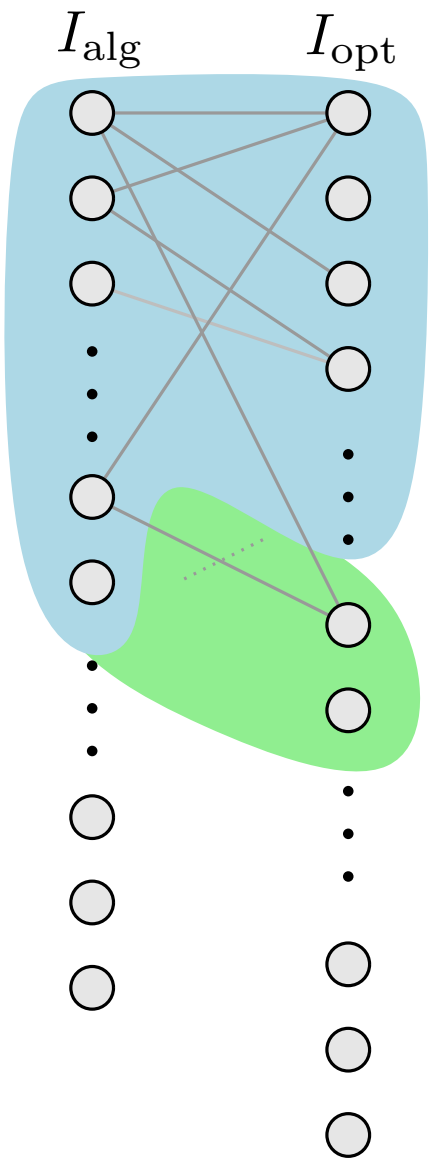
Proof sketch.



1. take clique-based separator S

SAS for graphs with small clique-based separators

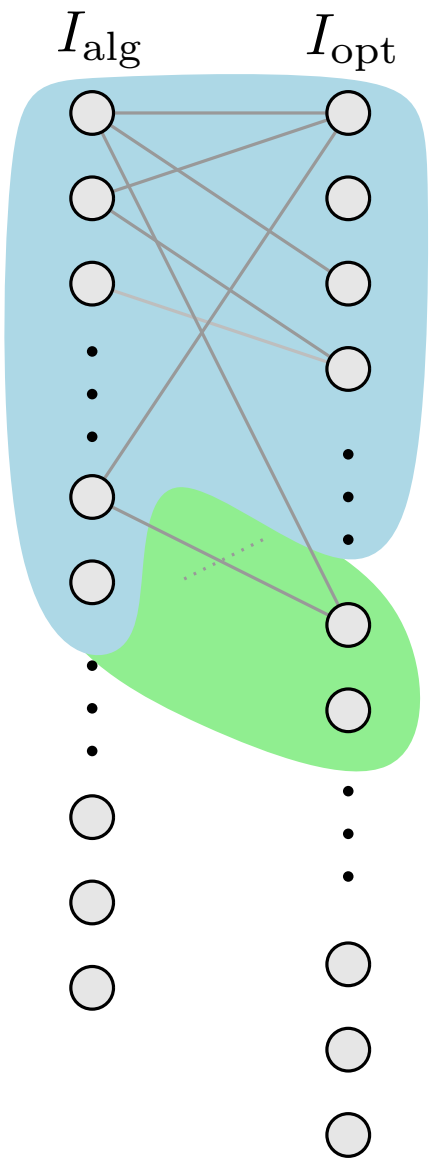
Proof sketch.



1. take clique-based separator S
2. recurse on half A where ratio $\frac{|I_{\text{opt}} \cap A|}{|I_{\text{alg}} \cap A|}$ is largest but add $S \cap I_{\text{alg}}$

SAS for graphs with small clique-based separators

Proof sketch.



1. take clique-based separator S
2. recurse on half A where ratio $\frac{|I_{\text{opt}} \cap A|}{|I_{\text{alg}} \cap A|}$ is largest but add $S \cap I_{\text{alg}}$

until $|I_{\text{opt}}| + |I_{\text{alg}}| = \Omega((1/\varepsilon)^{\frac{1}{1-\delta}})$

we still have $\frac{|I_{\text{opt}}|}{|I_{\text{alg}}|} \geqslant \frac{1}{1-\varepsilon/2}$

SAS for graphs with small clique-based separators

Theorem. There is a SAS for disk graphs (and other graphs with sublinear clique-based separators) for Independent Set.

Consider graph class with clique-based separators of weight $O(n^\delta)$, for $\delta < 1$.

Algorithm

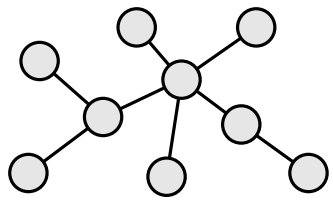
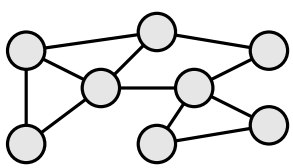
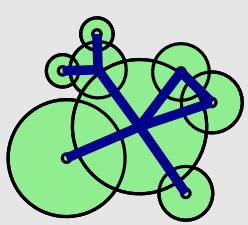
▷ Upon each insertion or deletion, do the following

1. **if** $|I_{\text{alg}}| \geqslant (1-\varepsilon) \cdot \text{Opt}$
2. **then** do nothing
3. **else** find $J_{\text{out}}, J_{\text{in}}$ such that
   - $(I_{\text{alg}} \setminus J_{\text{out}}) \cup J_{\text{in}}$ is independent set
   - $|J_{\text{in}}| > |J_{\text{out}}|$
   - $|J_{\text{out}}| + |J_{\text{in}}| = O((1/\varepsilon)^{\frac{1}{1-\delta}})$

   always possible!
4. replace J_{out} by J_{in} $\implies$ will restore $|I_{\text{alg}}| \geqslant (1-\varepsilon) \cdot \text{Opt}$

Stable Approximation Schemes for Independent Set?

Is there a SAS for Independent Set on

- trees?  **YES**
- graphs of maximum bounded degree? **NO**
- planar graphs?  **YES**
- disk graphs?  **YES** 

Concluding remarks

What we have seen

- stable algorithms and Stable Approximation Schemes (SAS)
- SAS results and no-SAS results for the broadcast range-assignment problem
- SAS results and no-SAS results for Independent Set
- advertisement for clique-based separators

What we have not seen

- algorithms with very small satbility parameter
  (see our exisiting and upcoming paper)

What is open

- many things!

Thanks for your attention!

and to Arpan Sadhukan and Frits Spieksma

TU/e

NET WORKS