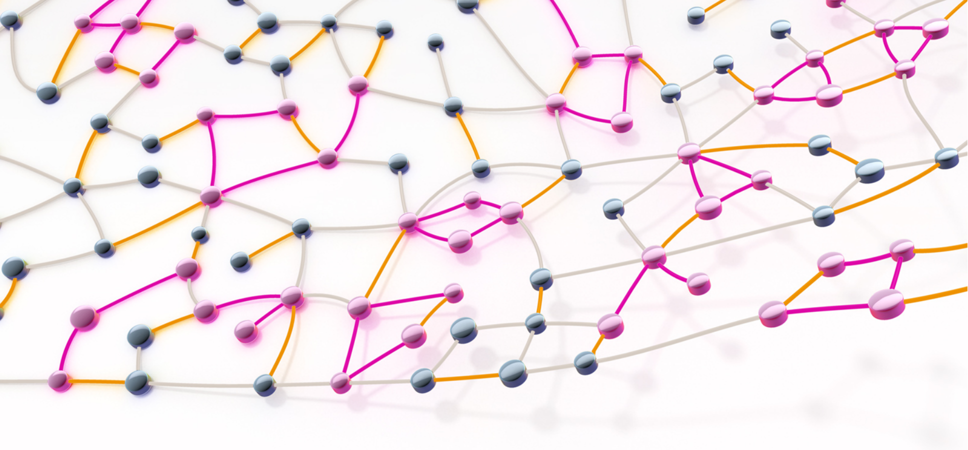
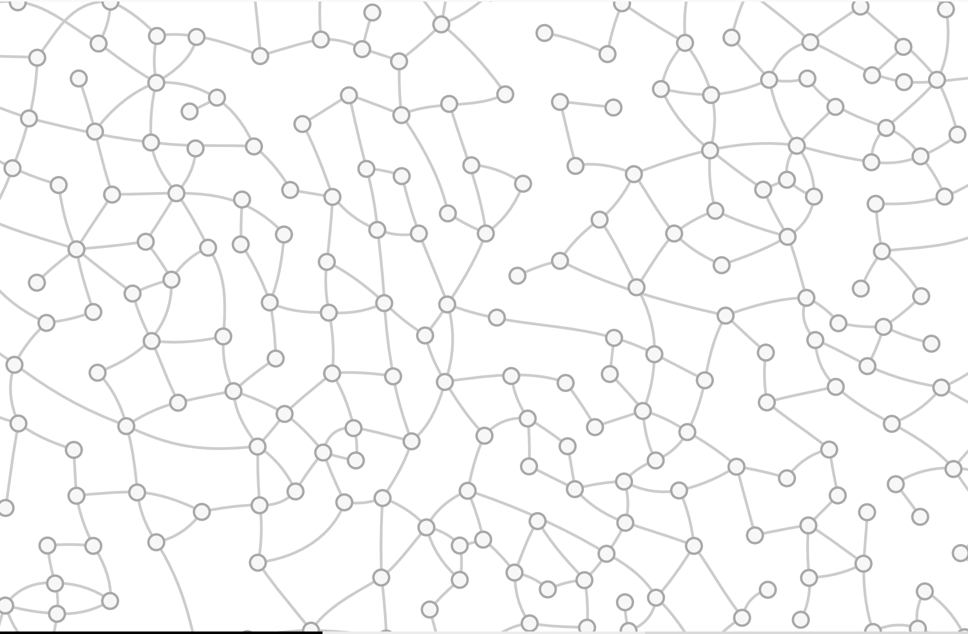


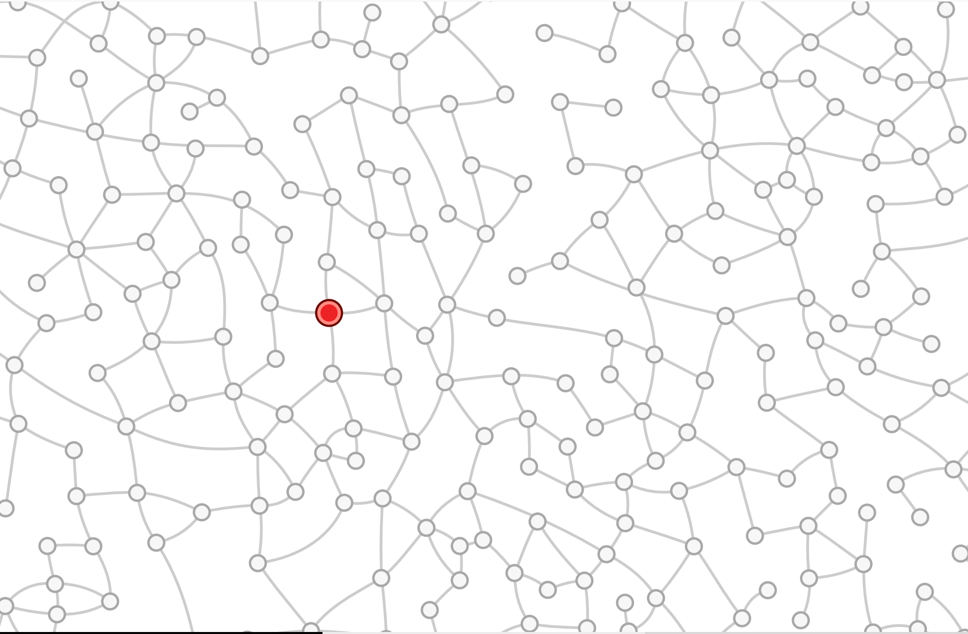
Bipartite Vertex Cover

Mika Göös University of Toronto & HIIT
Jukka Suomela University of Helsinki & HIIT

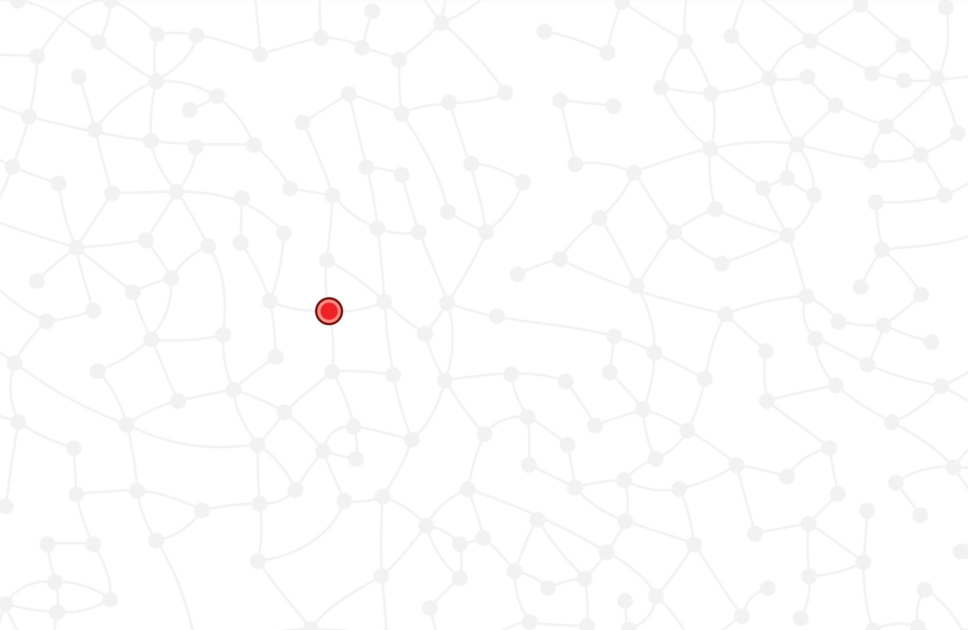
$\mathcal{LOCAL}$ model



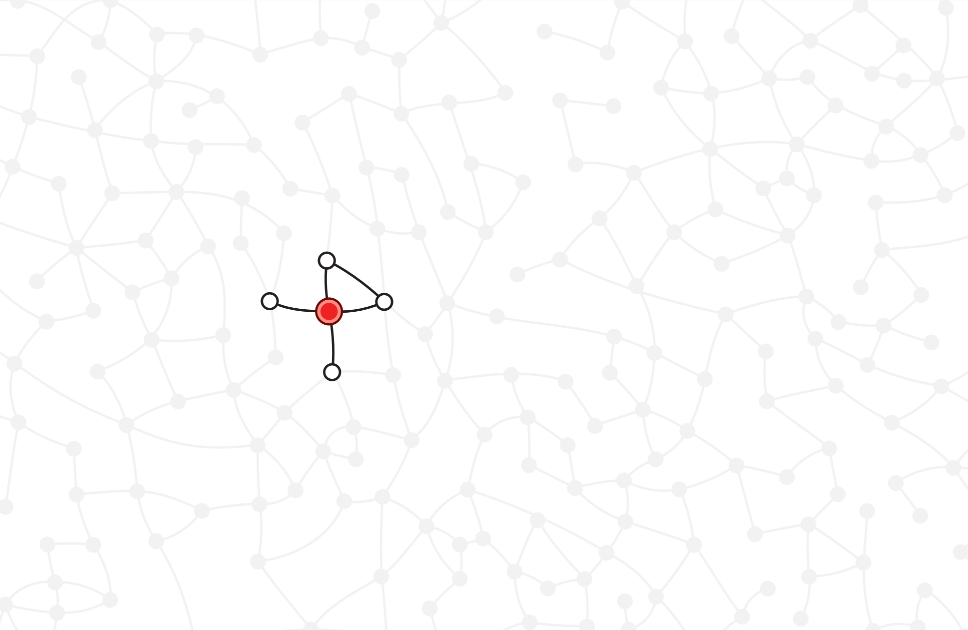
$\mathcal{LOCAL}$ model



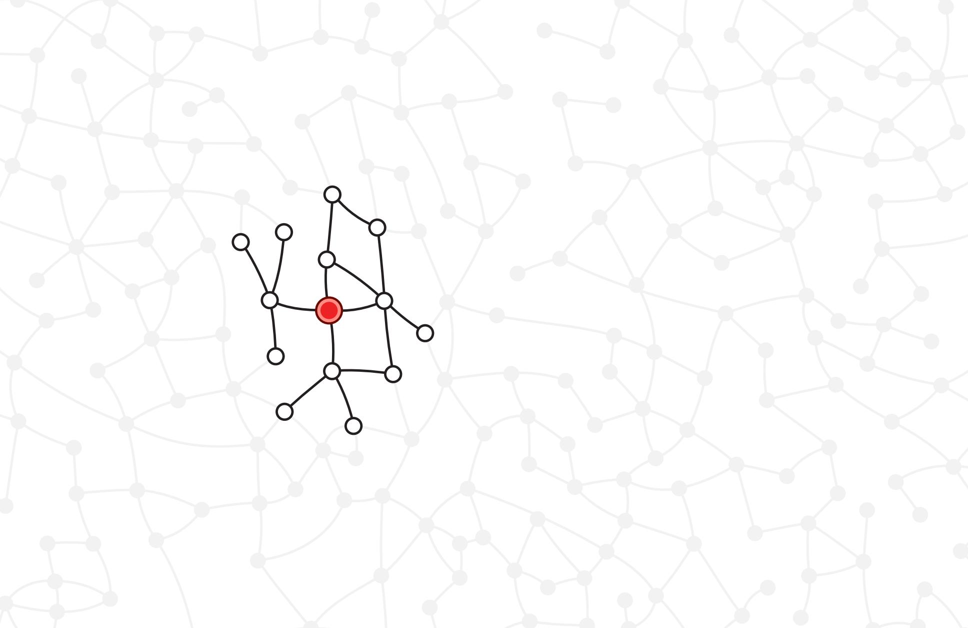
$\mathcal{LOCAL}$ model



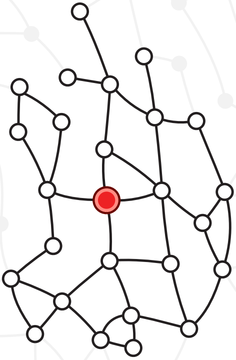
$\mathcal{LOCAL}$ model



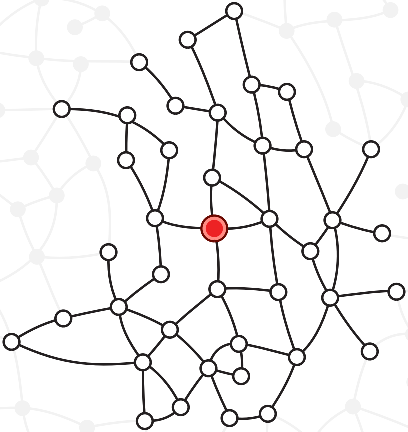
$\mathcal{LOCAL}$ model



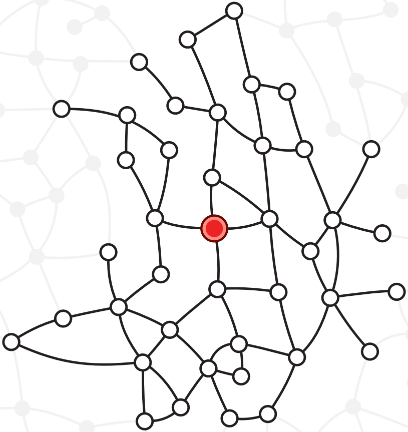
$\mathcal{LOCAL}$ model



$\mathcal{LOCAL}$ model

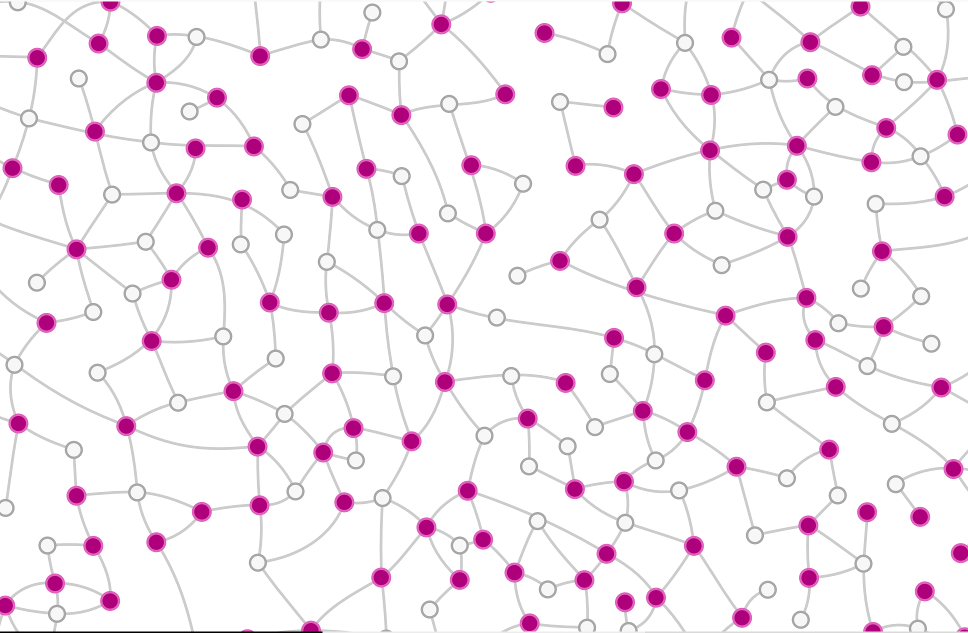


$\mathcal{LOCAL}$ model



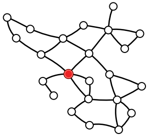
$\mapsto \{0, 1\}$

$\mathcal{LOCAL}$ model



$\mathcal{LOCAL}$ model

Definition:

$\mathbf{A}$: {  } $\to \{0, 1\}$

Run-time R
= radius-R neighbourhood:

1. Nodes have **unique IDs**
2. Nodes get **random strings** as input

Prior work on **Min Vertex Cover** (MIN-VC)

	Apx ratio	**Run-time**	
General graphs	$O(1)$	$\Omega(\sqrt{\log n})$	[KMW **PODC'04**]

Prior work on **Min Vertex Cover** (MIN-VC)

	Apx ratio	Run-time	
General graphs	$O(1)$	$\Omega(\sqrt{\log n})$	[KMW **PODC'04**]
Bounded degree	$O(1)$	0	
	$2+\epsilon$	$O_\epsilon(1)$	[KMW **SODA'06**]
	2	$O(1)$	[ÅS **SPAA'10**]
	$2-\epsilon$	$\Omega(\log n)$	[PR **'07**]

Prior work on **Min Vertex Cover** (MIN-VC)

	Apx ratio	**Run-time**	
General graphs	$O(1)$	$\Omega(\sqrt{\log n})$	[KMW **PODC'04**]
Bounded degree	$O(1)$	0	
	$2+\epsilon$	$O_\epsilon(1)$	[KMW **SODA'06**]
	2	$O(1)$	[ÅS **SPAA'10**]
	$2-\epsilon$	$\Omega(\log n)$	[PR **'07**]

Note: MIN-VC is solvable on **bipartite** graphs using sequential polynomial-time algorithms!

The **bipartite** case

Question: Can we approximate MIN-VC fast on **bipartite** graphs?

The **bipartite** case

Question: Can we approximate MIN-VC fast on **bipartite** graphs?

$(1+\epsilon)$-approximation scheme?

The **bipartite** case

Setting:
- Bipartite **2-coloured** graph
- Bounded degree $\Delta = O(1)$
- Compute $(1+\epsilon)$-approximation

The **bipartite** case

Setting:

– Bipartite **2-coloured** graph
– Bounded degree $\Delta = O(1)$
– Compute $(1+\epsilon)$-approximation

	Covering
Integer	**Min** VC

The **bipartite** case

Setting:
- Bipartite **2-coloured** graph
- Bounded degree $\Delta = O(1)$
- Compute $(1+\epsilon)$-approximation

	Covering
Integer	**Min** VC
LP	**Min** Frac. VC

The **bipartite** case

Setting:

- Bipartite **2-coloured** graph
- Bounded degree $\Delta = O(1)$
- Compute $(1+\epsilon)$-approximation

	Covering	Packing
Integer	**Min** VC	**Max** Matching
LP	**Min** Frac. VC	**Max** Frac. Matching

The **bipartite** case

Setting:
- Bipartite **2-coloured** graph
- Bounded degree $\Delta = O(1)$
- Compute $(1+\epsilon)$-approximation

	Covering		Packing
Integer	**Min** VC		**Max** Matching
LP	**Min** Frac. VC	=	**Max** Frac. Matching

= LP duality

The **bipartite** case

Setting:
- Bipartite **2-coloured** graph
- Bounded degree $\Delta = O(1)$
- Compute $(1+\epsilon)$-approximation

	Covering	Packing
Integer	**Min** VC	**Max** Matching
	$\parallel$	$\parallel$
LP	**Min** Frac. VC $=$	**Max** Frac. Matching

= LP duality
= Total unimodularity

The **bipartite** case

Setting:
– Bipartite **2-coloured** graph
– Bounded degree $\Delta = O(1)$
– Compute $(1+\epsilon)$-approximation

	Covering		Packing
Integer	**Min** VC	=	**Max** Matching
	‖		‖
LP	**Min** Frac. VC	=	**Max** Frac. Matching

= LP duality
= Total unimodularity
= König's theorem

The **bipartite** case

Setting:

- Bipartite **2-coloured** graph
- Bounded degree $\Delta = O(1)$
- Compute $(1+\epsilon)$-approximation

	Covering	**Packing**
Integer	**Min** VC	**Max** Matching
LP	$O_\epsilon(1)$	$O_\epsilon(1)$

[KMW **SODA'06**]

The **bipartite** case

Setting:
- Bipartite **2-coloured** graph
- Bounded degree $\Delta = O(1)$
- Compute $(1+\epsilon)$-approximation

	Covering	**Packing**
Integer	**Min** VC	$O_\epsilon(1)$
LP	$O_\epsilon(1)$	$O_\epsilon(1)$

[KMW **SODA'06**]
[NO **FOCS'08**], [ÅPRSU **'10**]

The **bipartite** case

	Covering	**Packing**
Integer	**???**	$O_\epsilon(1)$
LP	$O_\epsilon(1)$	$O_\epsilon(1)$

The **bipartite** case

	Covering	Packing
Integer	$\Omega(\log n)$	$O_\epsilon(1)$
LP	$O_\epsilon(1)$	$O_\epsilon(1)$

The **bipartite** case

Surprise: **No Sublogarithmic-Time Approximation Scheme for Bipartite Vertex Cover!**

	Covering	**Packing**
Integer	$\Omega(\log n)$	$O_\epsilon(1)$
LP	$O_\epsilon(1)$	$O_\epsilon(1)$

Our result

Main Theorem

$\exists\delta > 0$: No $o(\log n)$-time algorithm to $(1+\delta)$-approximate MIN-VC on 2-coloured graphs of max degree $\Delta = 3$

Our result

Main Theorem

$\exists\delta > 0$**:** No $o(\log n)$-time algorithm to $(1+\delta)$-approximate MIN-VC on 2-coloured graphs of max degree $\Delta = 3$

Lower bound is tight

1. There is $O_\epsilon(\log n)$-time approx. scheme [LS **'93**]
2. If $\Delta = 2$ there is $O_\epsilon(1)$-time approx. scheme

Why is **MIN-VC** difficult for distributed graph algorithms?

Short answer: Solving MIN-VC requires solving a hard **cut minimisation** problem

Why is MIN-VC difficult for distributed graph algorithms?

Short answer: Solving MIN-VC requires solving a hard **cut minimisation** problem

Strategy:
1. Reduce cut problem to MIN-VC
2. Prove that cut problem is hard

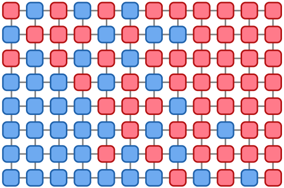
Reduction formalised

RECUT problem

Input: Labelled graph (G, ℓ_{in}) where $\ell_{\mathsf{in}} : V \to \{\mathsf{red}, \mathsf{blue}\}$
Output: Labelling $\ell_{\mathsf{out}} : V \to \{\mathsf{red}, \mathsf{blue}\}$ that minimises the size of the cut $|\ell_{\mathsf{out}}|$ subject to

- If ℓ_{in} is all-red then ℓ_{out} is all-red
- If ℓ_{in} is all-blue then ℓ_{out} is all-blue

ℓ_{in}



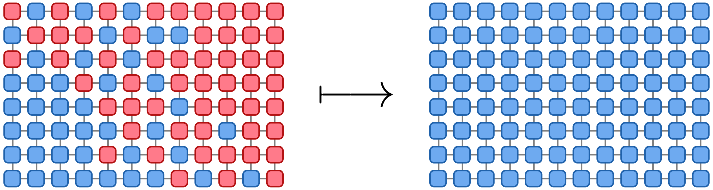
Reduction formalised

RECUT problem

Input: Labelled graph (G, ℓ_{in}) where $\ell_{\text{in}} : V \to \{\text{red}, \text{blue}\}$
Output: Labelling $\ell_{\text{out}} : V \to \{\text{red}, \text{blue}\}$ that minimises the size of the cut $|\ell_{\text{out}}|$ subject to

- If ℓ_{in} is all-red then ℓ_{out} is all-red
- If ℓ_{in} is all-blue then ℓ_{out} is all-blue

ℓ_{in} $\longmapsto$ Global optimum

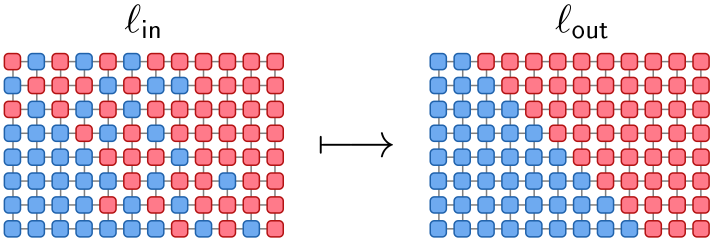


Reduction formalised

RECUT problem

Input: Labelled graph (G, ℓ_{in}) where $\ell_{\mathsf{in}} : V \to \{\mathsf{red}, \mathsf{blue}\}$
Output: Labelling $\ell_{\mathsf{out}} : V \to \{\mathsf{red}, \mathsf{blue}\}$ that minimises the size of the cut $|\ell_{\mathsf{out}}|$ subject to

- If ℓ_{in} is all-red then ℓ_{out} is all-red
- If ℓ_{in} is all-blue then ℓ_{out} is all-blue



Reduction formalised

RECUT problem

Input: Labelled graph (G, ℓ_{in}) where $\ell_{\mathsf{in}} : V \to \{\mathsf{red}, \mathsf{blue}\}$

Output: Labelling $\ell_{\mathsf{out}} : V \to \{\mathsf{red}, \mathsf{blue}\}$ that minimises the size of the cut $|\ell_{\mathsf{out}}|$ subject to

- If ℓ_{in} is all-red then ℓ_{out} is all-red
- If ℓ_{in} is all-blue then ℓ_{out} is all-blue

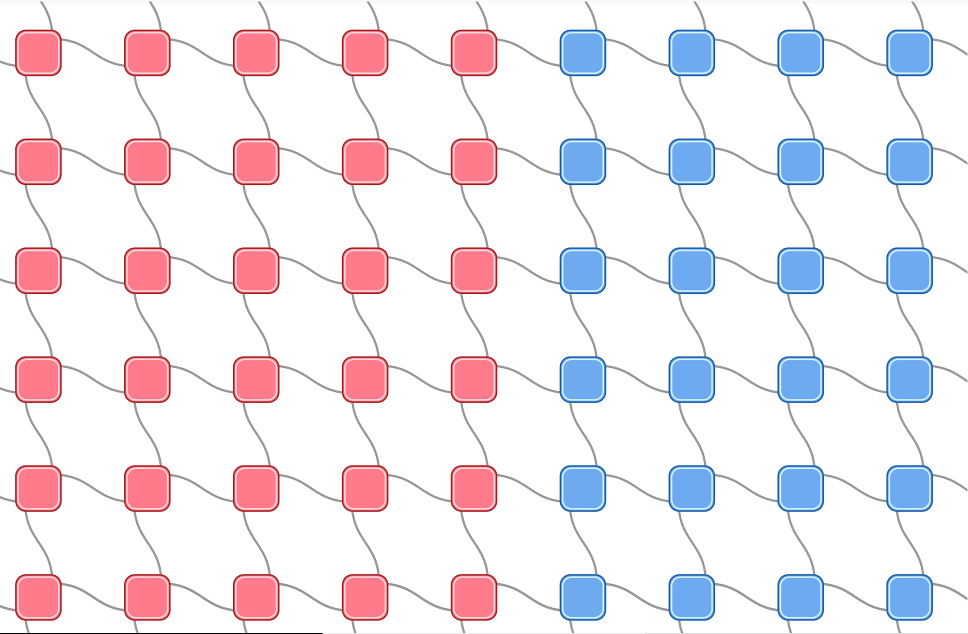
RECUT $\leq$ MIN-VC

If: MIN-VC can be $(1 + \epsilon)$-approximated in time R

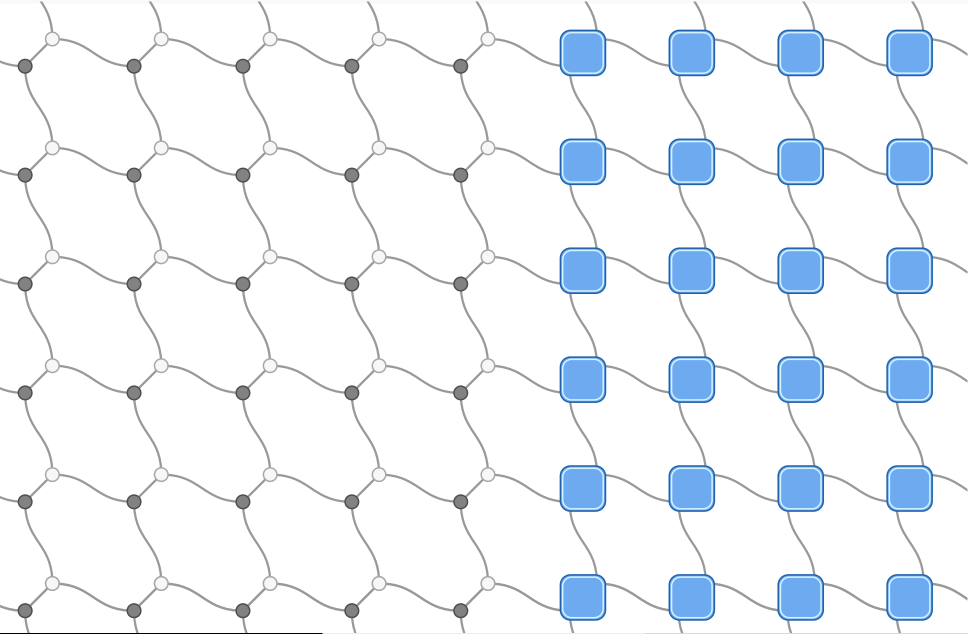
Then: We can compute in time R a RECUT of density

$$\frac{|\ell_{\mathsf{out}}|}{|E|} \leq \epsilon$$

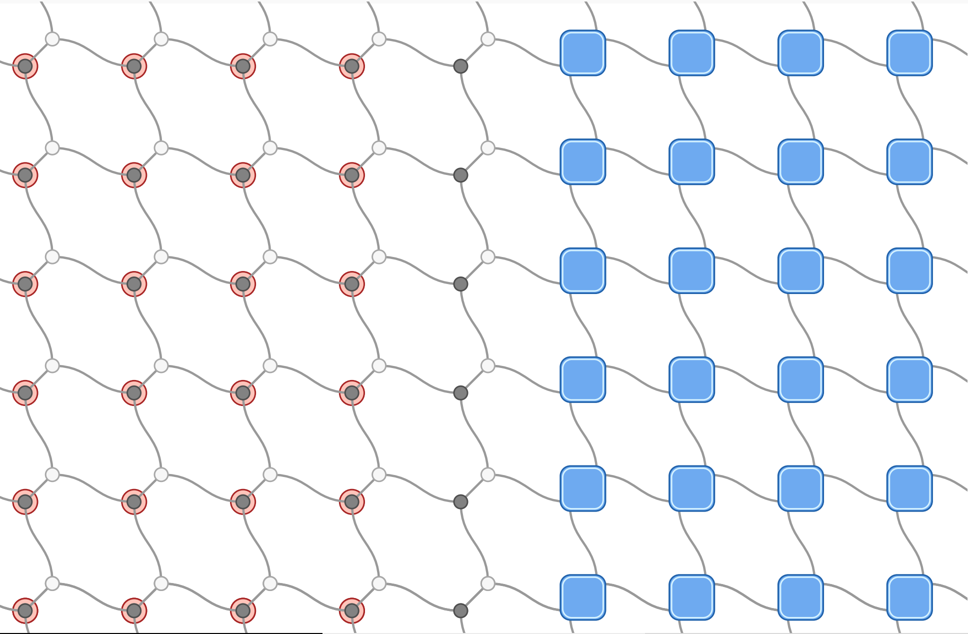
Reduction in pictures



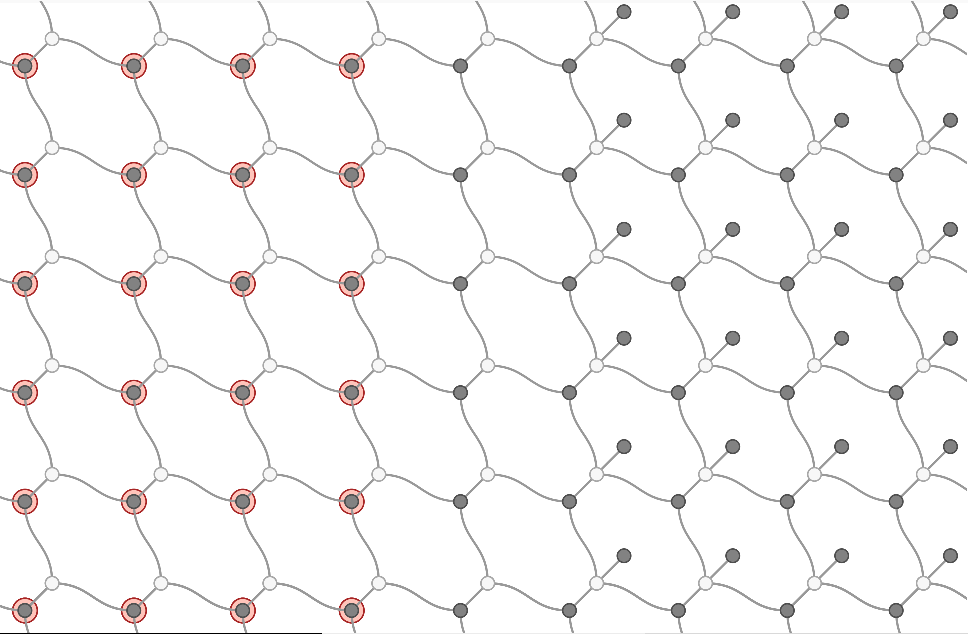
Reduction in pictures



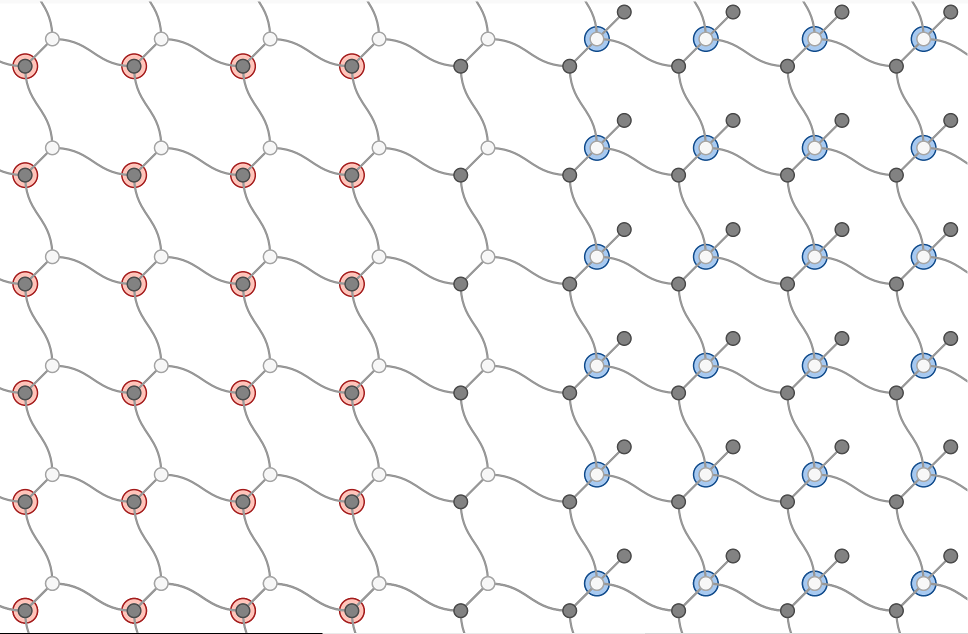
Reduction in pictures



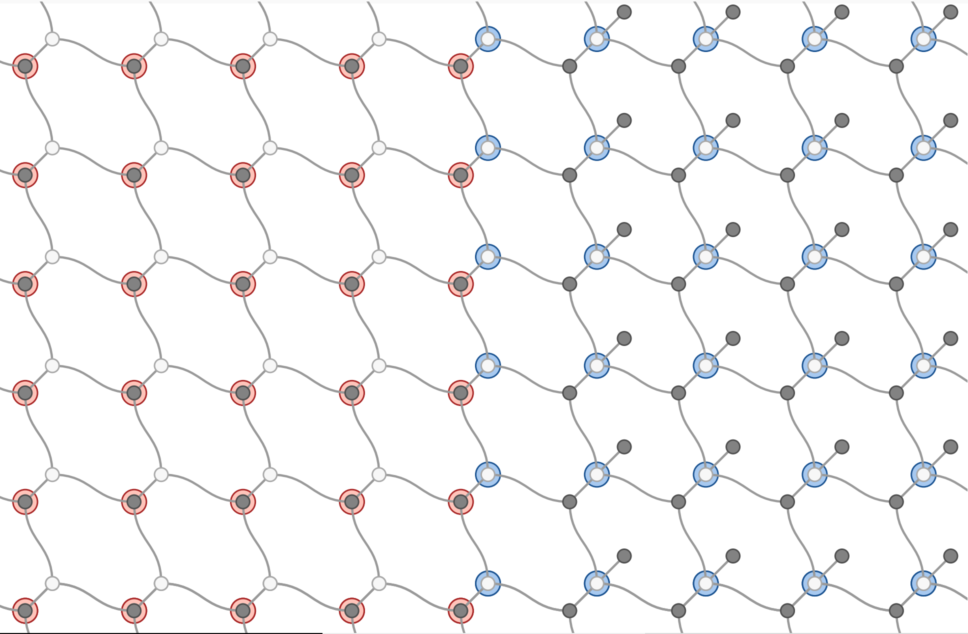
Reduction in pictures



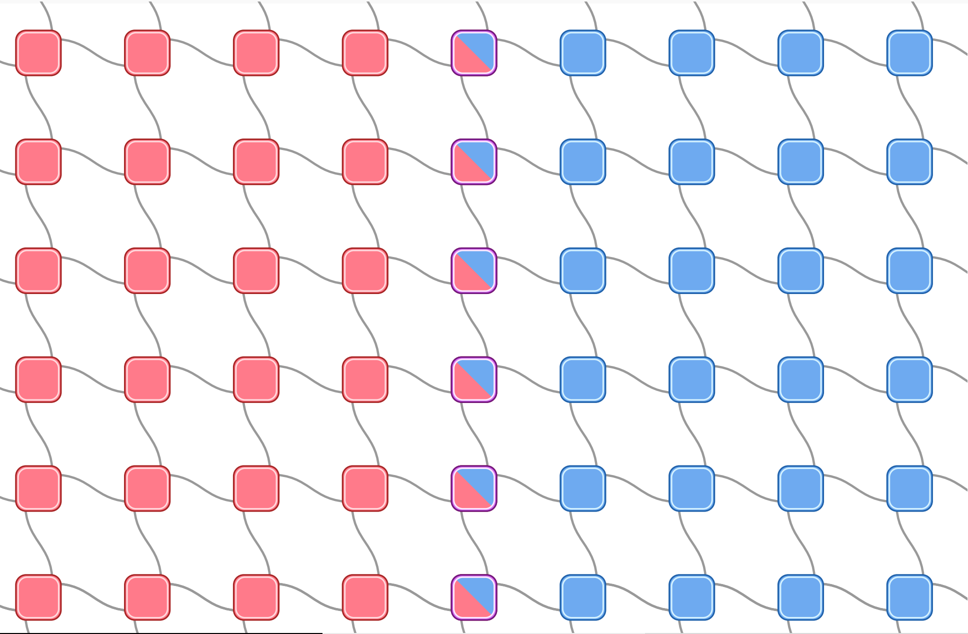
Reduction in pictures



Reduction in pictures



Reduction in pictures



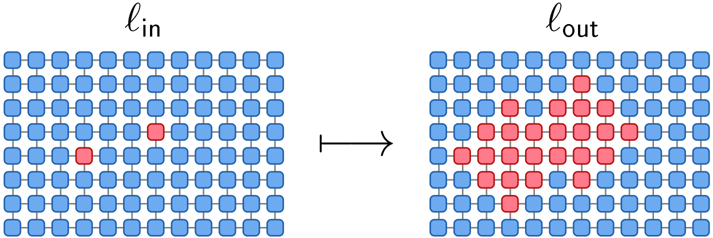
On Recut

Theorem: Recut $\leq$ Min-VC

On RECUT

Theorem: $\textsc{Recut} \le \textsc{Min-VC}$

Sometimes RECUT Is Easy:
The algorithm "Output red iff you see any red nodes" computes a small RECUT on grid-like graphs



On Recut

Theorem: Recut $\leq$ Min-VC

Sometimes Recut Is Easy:
The algorithm "Output red iff you see any red nodes" computes a small Recut on grid-like graphs

Therefore: We consider **expander graphs** that satisfy

$$|\ell| \geq \delta \cdot \min(|\ell^{-1}(\mathsf{red})|, |\ell^{-1}(\mathsf{blue})|)$$

On RECUT

Theorem: RECUT $\leq$ MIN-VC

Sometimes RECUT Is Easy:
The algorithm "Output red iff you see any red nodes" computes a small RECUT on grid-like graphs

Therefore: We consider **expander graphs** that satisfy

$$|\ell| \geq \delta \cdot \min(|\ell^{-1}(\mathsf{red})|, |\ell^{-1}(\mathsf{blue})|)$$

Main Technical Lemma: We fool a fast algorithm into producing a **balanced** RECUT

$$|\ell_{\mathsf{out}}^{-1}(\mathsf{red})| \approx |\ell_{\mathsf{out}}^{-1}(\mathsf{blue})| \approx n/2$$

Conclusions

Our result:

- No $o(\log n)$-time approximation scheme for **Min-VC** on 2-coloured graphs with $\Delta = 3$

Conclusions

Our result:

- No $o(\log n)$-time approximation scheme for **MIN-VC** on 2-coloured graphs with $\Delta = 3$

Open problems:

- Approximation ratios for $O(1)$-time algorithms?
- Derandomising Linial–Saks requires designing deterministic algorithms for RECUT

Conclusions

Our result:

- No $o(\log n)$-time approximation scheme for **Min-VC** on 2-coloured graphs with $\Delta = 3$

Open problems:

- Approximation ratios for $O(1)$-time algorithms?
- Derandomising Linial–Saks requires designing deterministic algorithms for Recut

Cheers!