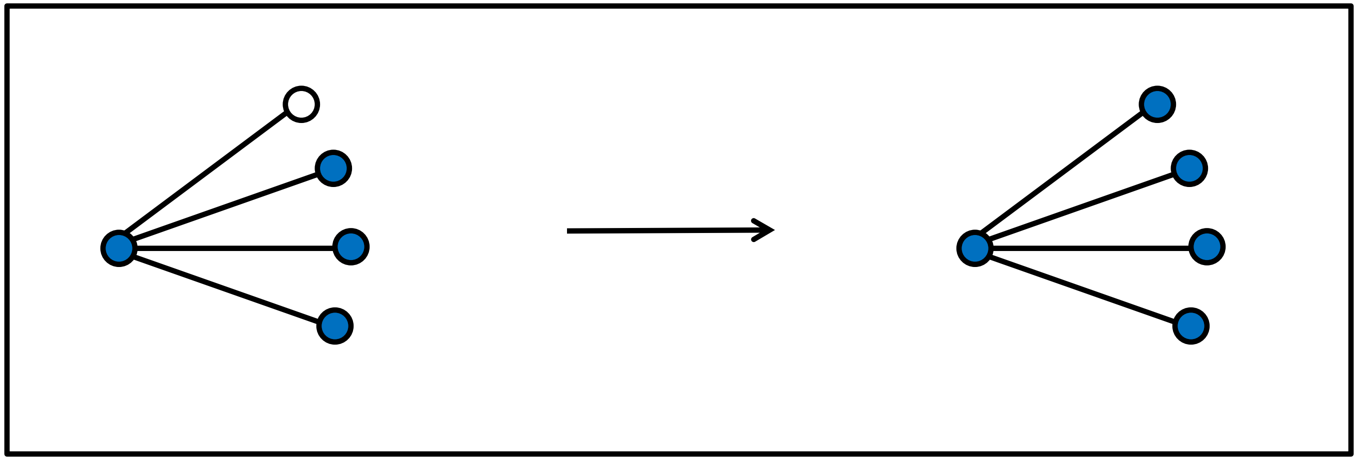
Computational methods and heuristics for zero forcing

Boris Brimkov

JMM 2021

Slippery Rock University
Mathematics and Statistics

Zero forcing



Blue vertex with exactly one white neighbor turns that neighbor blue

Related problems

- Power network monitoring (Haynes et al. 2002)
- Target set selection (Kempe et al. 2003)
- Zero forcing / minimum rank (AIM Group 2008)
- Quantum control (Burgarth et al. 2007)
- Fast mixed network searching (Yang 2013)
- k-forcing (Amos et al. 2015)
- Connected power domination (Brimkov et al. 2018)
- Positive semidefinite zero forcing (Ekstrand et al. 2014)
- Probabilistic zero forcing (Kang & Yi 2012)

Related problems

- Power network monitoring (Haynes et al. 2002)
- Target set selection (Kempe et al. 2003)
- Zero forcing / minimum rank (AIM Group 2008)
- Quantum control (Burgarth et al. 2007)
- Fast mixed network searching (Yang 2013)
- k-forcing (Amos et al. 2015)
- Connected power domination (Brimkov et al. 2018)
- Positive semidefinite zero forcing (Ekstrand et al. 2014)
- Probabilistic zero forcing (Kang & Yi 2012)

All of these problems are computationally difficult

Computational methods

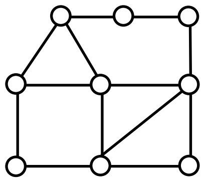
- Guess-and-check / brute force (Everyone)
- Dynamic programming (Butler et al. 2014)
- Integer programming and branch-and-bound (Aazami 2010, Ackerman et al. 2010, Ben-Zwi et al. 2011, Mahaei and Hagh 2012, Chiang et al. 2013, Agra et al. 2019)
- Boolean Satisfiability (Brimkov et al. 2020)
- Heuristics (Agra et al. 2019, Brimkov et al. 2020)

Dynamic programming

Wavefront Algorithm

Data: Graph $G = (V, E)$
Result: Zero forcing number of G
$\mathcal{C} \leftarrow \{(\emptyset, 0)\}$;
for $R \in [n]$ **do**
  for $(S, r) \in \mathcal{C}$ **do**
    for $v \in V$ **do**
      $S' \leftarrow cl(S \cup N[v])$;
      $r' \leftarrow r + |\{v\} \setminus S| + \max\{|N(v) \setminus S| - 1, 0\}$;
      if $r' \leq R$ **and** $(S', i) \notin \mathcal{C}$ **for** $i \leq R$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{(S', r')\}$;
        if $S' = V$ **then return** r';

Dynamic programming



Wavefront Algorithm

Data: Graph $G = (V, E)$

Result: Zero forcing number of G

$\mathcal{C} \leftarrow \{(\emptyset, 0)\}$;

for $R \in [n]$ **do**

    for $(S, r) \in \mathcal{C}$ **do**

        for $v \in V$ **do**

            $S' \leftarrow cl(S \cup N[v])$;

            $r' \leftarrow r + |\{v\} \backslash S| + \max\{|N(v) \backslash S| - 1, 0\}$;

            if $r' \leq R$ **and** $(S', i) \notin \mathcal{C}$ **for** $i \leq R$ **then**

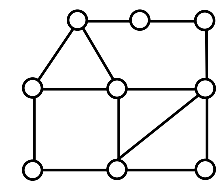
                $\mathcal{C} \leftarrow \mathcal{C} \cup \{(S', r')\}$;

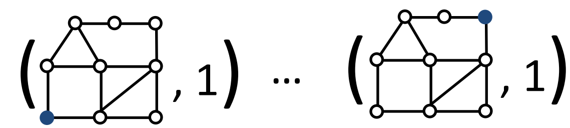
                if $S' = V$ **then return** r';

Dynamic programming

Wavefront Algorithm

Data: Graph $G = (V, E)$
Result: Zero forcing number of G
$\mathcal{C} \leftarrow \{(\emptyset, 0)\}$;
for $R \in [n]$ **do**
  for $(S, r) \in \mathcal{C}$ **do**
    for $v \in V$ **do**
      $S' \leftarrow cl(S \cup N[v])$;
      $r' \leftarrow r + |\{v\} \backslash S| + \max\{|N(v) \backslash S| - 1, 0\}$;
      if $r' \leq R$ **and** $(S', i) \notin \mathcal{C}$ **for** $i \leq R$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{(S', r')\}$;
        if $S' = V$ **then return** r';

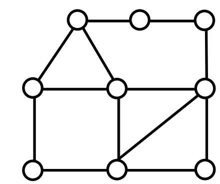


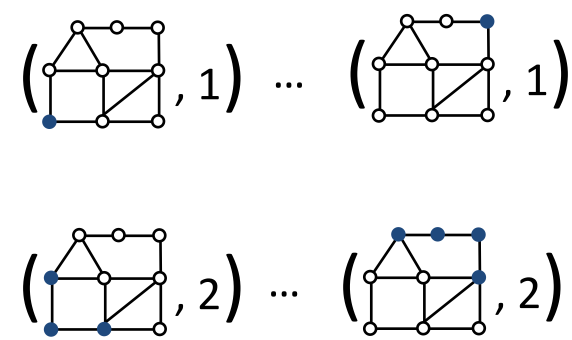


Dynamic programming

Wavefront Algorithm

Data: Graph $G = (V, E)$
Result: Zero forcing number of G
$\mathcal{C} \leftarrow \{(\emptyset, 0)\}$;
for $R \in [n]$ **do**
  for $(S, r) \in \mathcal{C}$ **do**
    for $v \in V$ **do**
      $S' \leftarrow cl(S \cup N[v])$;
      $r' \leftarrow r + |\{v\} \backslash S| + \max\{|N(v) \backslash S| - 1, 0\}$;
      if $r' \leq R$ **and** $(S', i) \notin \mathcal{C}$ **for** $i \leq R$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{(S', r')\}$;
        if $S' = V$ **then return** r';





Dynamic programming

Wavefront Algorithm

Data: Graph $G = (V, E)$

Result: Zero forcing number of G

$\mathcal{C} \leftarrow \{(\emptyset, 0)\}$;

for $R \in [n]$ **do**

  for $(S, r) \in \mathcal{C}$ **do**

    for $v \in V$ **do**

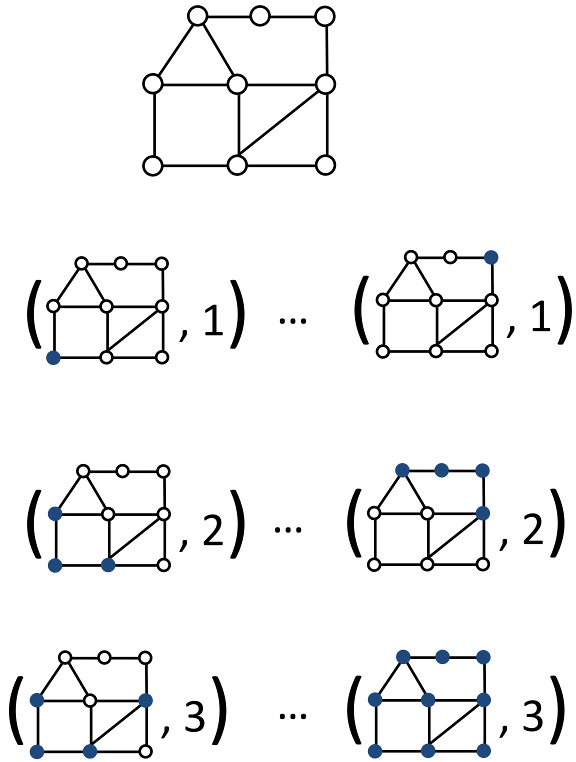
      $S' \leftarrow cl(S \cup N[v])$;

      $r' \leftarrow r + |\{v\} \backslash S| + \max\{|N(v) \backslash S| - 1, 0\}$;

      if $r' \leq R$ **and** $(S', i) \notin \mathcal{C}$ **for** $i \leq R$ **then**

        $\mathcal{C} \leftarrow \mathcal{C} \cup \{(S', r')\}$;

        if $S' = V$ **then return** r';



IP formulations

Compact formulation

$$
\begin{aligned}
\min \quad & \sum_{v \in V} s_v \\
\text{s.t.:} \quad & s_v + \sum_{e \in \delta^-(v)} y_e = 1 \qquad \forall v \in V \\
& x_u - x_v + (T+1) y_e \leq T \qquad \forall e = (u,v) \in E \\
& x_w - x_v + (T+1) y_e \leq T \qquad \forall e = (u,v) \in E, \forall w \in N(u) \backslash \{v\} \\
& x \in \{0, \ldots, T\}^n, s \in \{0,1\}^n, y \in \{0,1\}^m
\end{aligned}
$$

Noncompact formulation

$$
\begin{aligned}
\min \quad & \sum_{v \in V} s_v \\
\text{s.t.:} \quad & \sum_{v \in B} s_v \geq 1 \qquad \forall B \in \mathcal{B} \\
& s \in \{0,1\}^n
\end{aligned}
$$

IP formulations

Compact formulation

$$
\begin{aligned}
\min \quad & \sum_{v \in V} s_v \\
\text{s.t.:} \quad & s_v + \sum_{e \in \delta^-(v)} y_e = 1 && \forall v \in V \\
& x_u - x_v + (T+1) y_e \leq T && \forall e = (u,v) \in E \\
& x_w - x_v + (T+1) y_e \leq T && \forall e = (u,v) \in E, \forall w \in N(u) \setminus \{v\} \\
& x \in \{0, \dots, T\}^n, s \in \{0,1\}^n, y \in \{0,1\}^m
\end{aligned}
$$

Noncompact formulation

$$
\begin{aligned}
\min \quad & \sum_{v \in V} s_v \\
\text{s.t.:} \quad & \sum_{v \in B} s_v \geq 1 && \forall B \in \mathcal{B} \\
& s \in \{0,1\}^n
\end{aligned}
$$

$$
\begin{aligned}
\min \quad & \sum_{v \in V} x_v \\
\text{s.t.:} \quad & \sum_{v \in V} x_v \geq 1 \\
& x_w - x_v + \sum_{a \in N(w) \setminus \{v\}} x_a \geq 0 && \forall (v,w) \text{ with } v \in V, w \in N(v) \\
& x_v = 0 \qquad \forall v \in cl(S) && x \in \{0,1\}^n
\end{aligned}
$$

Boolean SAT formulation

$$\min \quad \sum_{v \in V} (\deg(v) z_v + s_v)$$

$$\text{s.t.:} \quad \bigwedge_{B \in \mathcal{B}} \left(\bigvee_{v \in B} s_v \vee \bigvee_{v \in T(B)} z_v \right) \wedge \left(\bigvee_{v \in V} z_v \right) \wedge \left(\bigwedge_{\substack{v \in V \\ w \in cl(N[v])}} (\neg\, s_w \vee \neg\, z_v) \right)$$

Boolean SAT formulation

$$\min \quad \sum_{v\in V}(\deg(v)z_v + s_v)$$

$$\text{s.t.:} \quad \bigwedge_{B\in\mathcal{B}}\left(\bigvee_{v\in B} s_v \vee \bigvee_{v\in T(B)} z_v\right) \wedge \left(\bigvee_{v\in V} z_v\right) \wedge \left(\bigwedge_{\substack{v\in V\\ w\in cl(N[v])}}(\neg\, s_w \vee \neg\, z_v)\right)$$

To generate violated constraints:

$$\min \quad \sum_{v\in V} x_v$$

$$\text{s.t.:} \quad \bigwedge_{\substack{v\in V\\ w\in N(v)}}\left(x_w \vee \neg\, x_v \vee \bigvee_{a\in N(w)\setminus\{v\}} x_a\right) \wedge \left(\bigvee_{v\in V} x_v\right) \wedge \left(\bigwedge_{v\in cl(S)} \neg\, x_v\right)$$

Alternate constraint generation

Algorithm 1: Greedy algorithm for finding minimal violated forts

1 **Input:** Graph $G = (V, E)$, set $S \subset V$ corresponding to solution of RMP;
2 **Output:** Minimal fort B which violates S;
3 $C \leftarrow cl(S)$;
4 **for** $v \in V \backslash C$ **do**
5   **if** $cl(C \cup \{v\}) \neq V$ **then**
6         $C \leftarrow C \cup \{v\}$;
7   **goto** line 4;
8 **return** $B \leftarrow V \backslash C$

Runtime comparisons

G	$\|V\|$	$Z(G)$	Wavefront	IP Models	Boolean
karate	34	13	329.10	0.16 (M)	**0.02**
chesapeake	39	14	10.91	35.43 (M)	**1.94**
dolphins	62	14	2405.56	246.24 (D)	**108.59**
lesmis	77	40	T	2708.95 (D)	**3.74**
polbooks	105	N/A	T	{14/27} (D)	{**18/27**}
adjnoun	112	N/A	T	{**9/30**} (M)	{9/32}
football	115	N/A	T	{**9/38**} (M)	{0/40}
jazz	198	N/A	T	{**27/96**} (M)	{10/102}
celegansneural	297	N/A	T	{**28/100**} (M)	{**15/89**}
IEEE 14	14	4	0.01	0.01 (M)	**0.001**
IEEE 24	24	6	0.07	0.01 (M)	**0.004**
IEEE 30	30	7	0.82	0.03 (M)	**0.010**
IEEE 39	39	7	6.69	0.04 (M)	**0.019**
IEEE 57	57	9	18.76	1.97 (M)	**0.282**
RTS 96	73	15	T	0.78 (M)	**0.641**
IEEE 118	118	26	T	734.96 (I)	**10.367**
IEEE 300	300	N/A	T	{**73/75**} (I)	{62/77}

Runtime comparisons

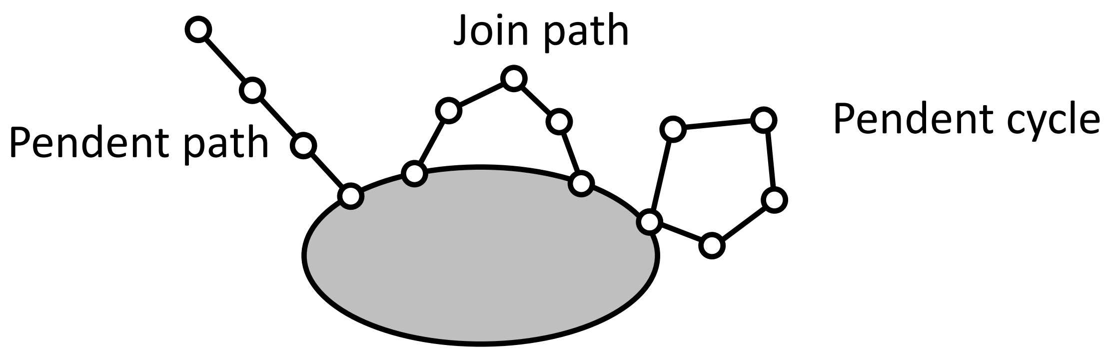
	$\lvert V\rvert$	$Z(G)$	Wavefront	IP Models	Boolean
Cubic graphs	10	3	0.01	0.01 (M)	**0.000**
	20	5	0.02	0.04 (M)	**0.009**
	30	6	**0.13**	0.42 (M)	0.142
	40	8	**1.95**	6.10 (E)	2.624
	50	9	**6.69**	30.90 (E)	17.527
	60	11	**156.83**	1363.05 (E)	3138.832
	70	12	**370.50**	2852.27 (E)	{10.8/12.4}
	80	N/A	**1615.85**	5341.15 (E)	{10.8/13.8}
	90	N/A	**3101.29**	{9.4/13.6} (F)	{9.6/14.4}
	100	N/A	T	{**9.6/15.4**} (F)	{9.6/15.8}
Avg-cubic	20	5.6	0.10	0.02 (M)	**0.003**
	40	10.0	796.89	0.63 (M)	**0.063**
	60	14.4	T	9.24 (I)	**0.867**
	80	19.4	T	15.45 (I)	**7.150**
	100	23.4	T	**37.23** (I)	345.232
	120	30.5	T	**109.13** (I)	382.478

Facets of zero forcing polytope

- If a polytope has a large number of facets, it is usually inherently difficult to solve with IP

Facets of zero forcing polytope

- If a polytope has a large number of facets, it is usually inherently difficult to solve with IP
- There can be $\Omega(1.3247^{n/3})$ facets defined by forts in pendant paths, join paths, and pendant cycles



Heuristics

- **Single vertex largest closure:** add a single white vertex to Z, so that the resulting closure is maximized
- **Neighborhood largest closure:** add a single white vertex and all-but-one of its white neighbors to Z, so that the resulting closure is maximized
- **Neighborhood scaled closure:** add a single white vertex and all-but-one of its white neighbors to Z, so that the *ratio* of the resulting closure to the number of vertices added is maximized

Closure based heuristics

Input: Graph $G = (V, E)$;
Output: Zero forcing set Z of G;
$Z \leftarrow \emptyset$;
$C \leftarrow \emptyset$;
while $C \neq V$ **do**
    $C^* \leftarrow \emptyset$;
    $A \leftarrow \emptyset$;
    for $v \in V \backslash C$ **do**
        $S \leftarrow C \cup A'(v, C)$;
        if $f(cl(S), v, C) > f(C^*, v, C)$ **then**
            $A \leftarrow A'(v, C)$;
            $C^* \leftarrow cl(S)$;
    $C \leftarrow C^*$;
    $Z \leftarrow Z \cup A$;
    for $v \in Z$ **do**
        if $cl(Z \setminus \{v\}) = V$ **then**
            $Z \leftarrow Z \setminus \{v\}$;
return Z

Single vertex largest closure

$$A'(v, C) = \{v\}$$
$$f(S, v, C) = |S|$$

Neighborhood largest closure

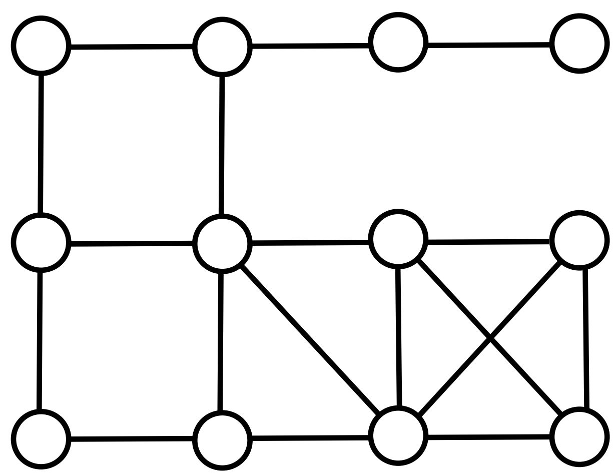
$$A'(v, C) = (N[v] \cap (V \backslash C)) \backslash \{u\}$$
for some $u \in N(v) \cap (V \backslash C)$
$$f(S, v, C) = |S|$$

Neighborhood scaled closure

$$A'(v, C) = (N[v] \cap (V \backslash C)) \backslash \{u\}$$
for some $u \in N(v) \cap (V \backslash C)$
$$f(S, v, C) = (|cl(S)| - |C|)/|A'(v, C)|$$

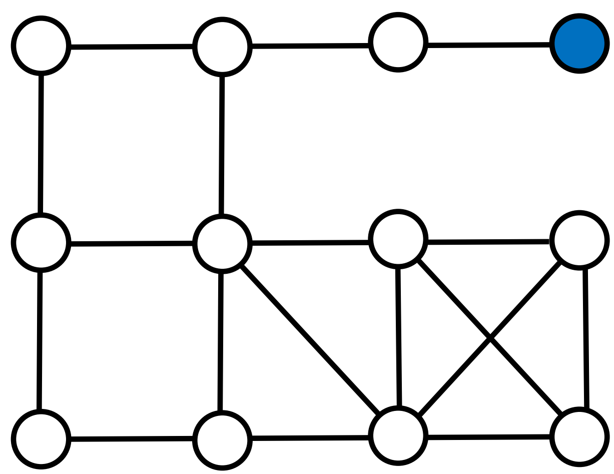
Heuristics

Single vertex largest closure: add a single white vertex to Z, so that the resulting closure is maximized



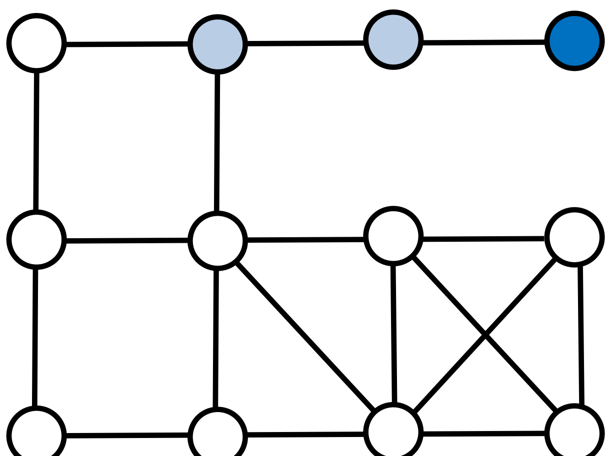
Heuristics

Single vertex largest closure: add a single white vertex to Z, so that the resulting closure is maximized



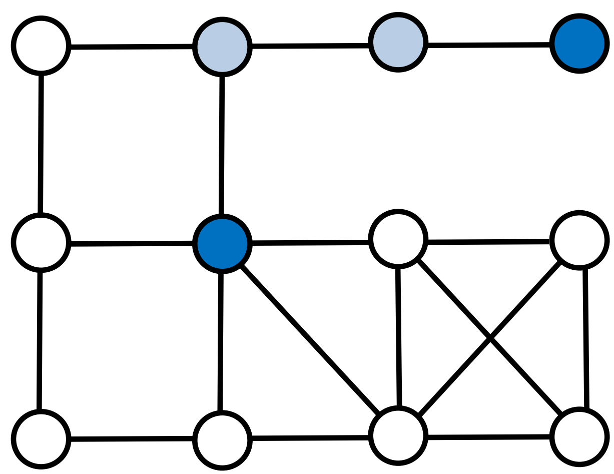
Heuristics

Single vertex largest closure: add a single white vertex to Z, so that the resulting closure is maximized



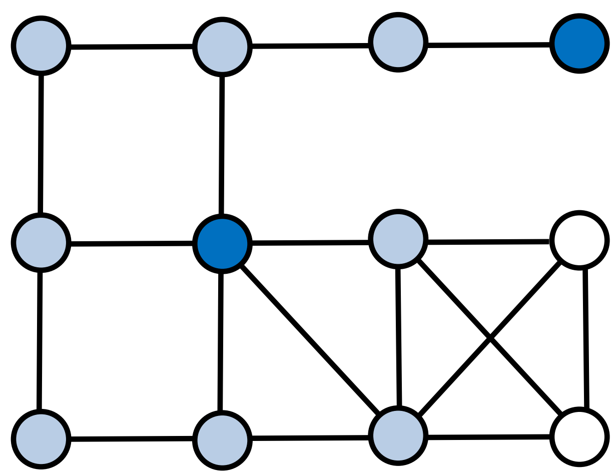
Heuristics

Single vertex largest closure: add a single white vertex to Z, so that the resulting closure is maximized



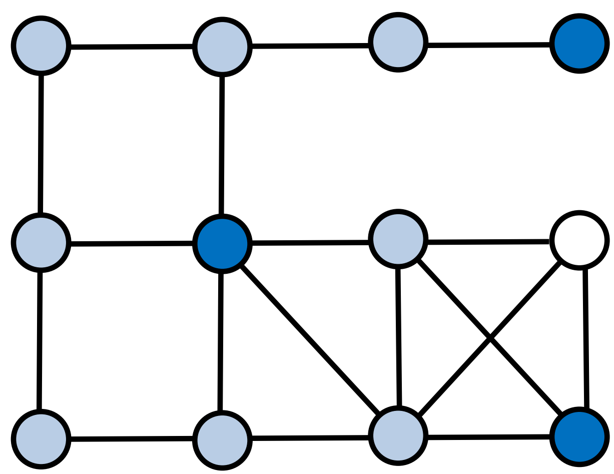
Heuristics

Single vertex largest closure: add a single white vertex to Z, so that the resulting closure is maximized



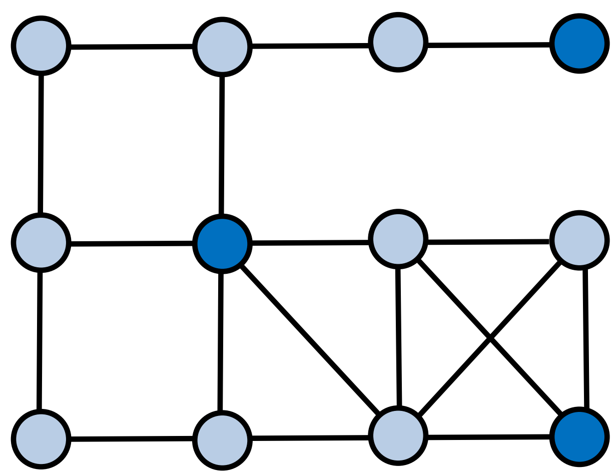
Heuristics

Single vertex largest closure: add a single white vertex to Z, so that the resulting closure is maximized



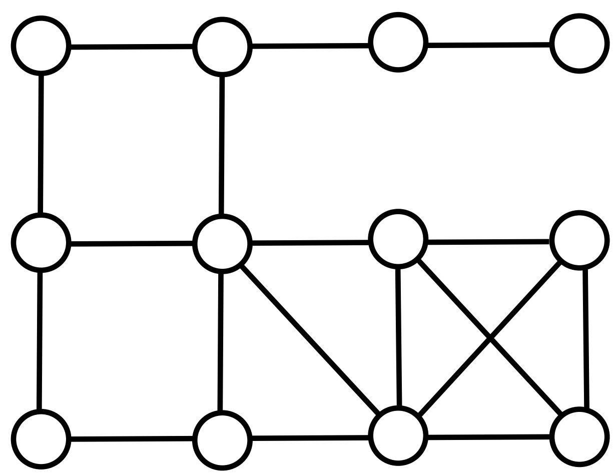
Heuristics

Single vertex largest closure: add a single white vertex to Z, so that the resulting closure is maximized



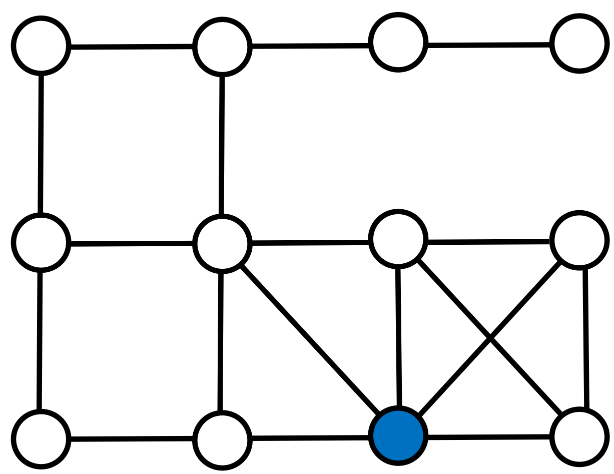
Heuristics

Neighborhood largest closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the resulting closure is maximized; take minimal subset



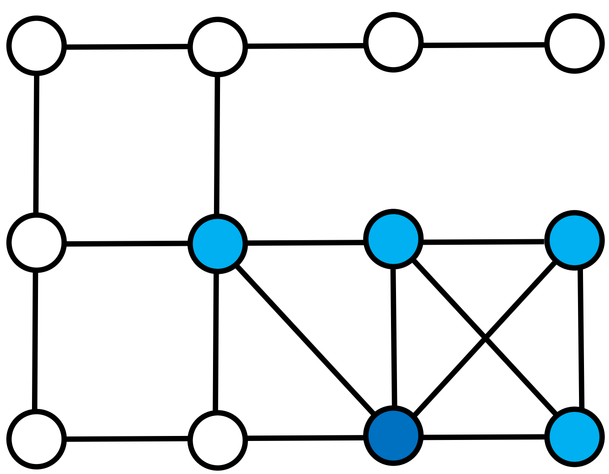
Heuristics

Neighborhood largest closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the resulting closure is maximized; take minimal subset



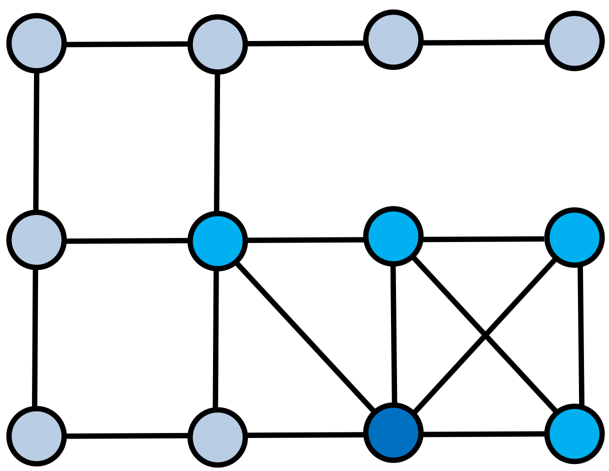
Heuristics

Neighborhood largest closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the resulting closure is maximized; take minimal subset



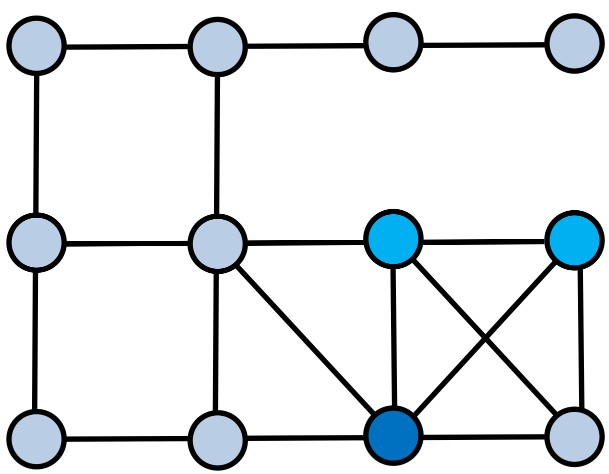
Heuristics

Neighborhood largest closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the resulting closure is maximized; take minimal subset



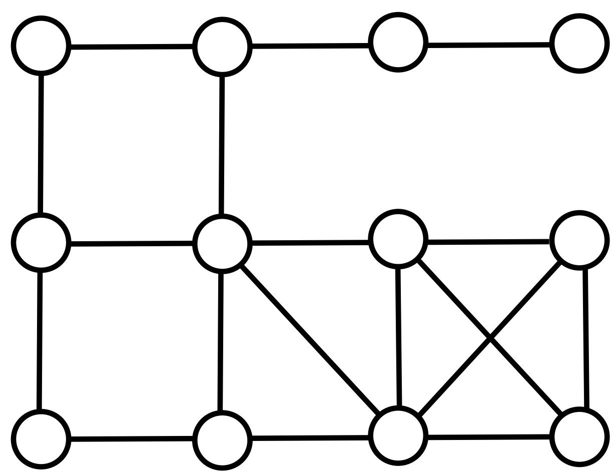
Heuristics

Neighborhood largest closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the resulting closure is maximized; take minimal subset



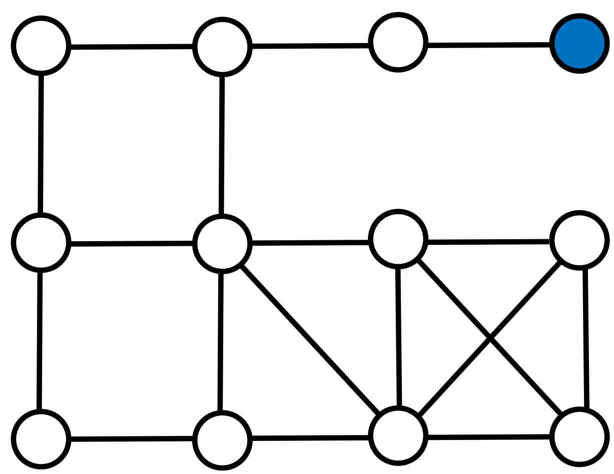
Heuristics

Neighborhood scaled closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the *ratio* of the resulting closure to the number of vertices added is maximized



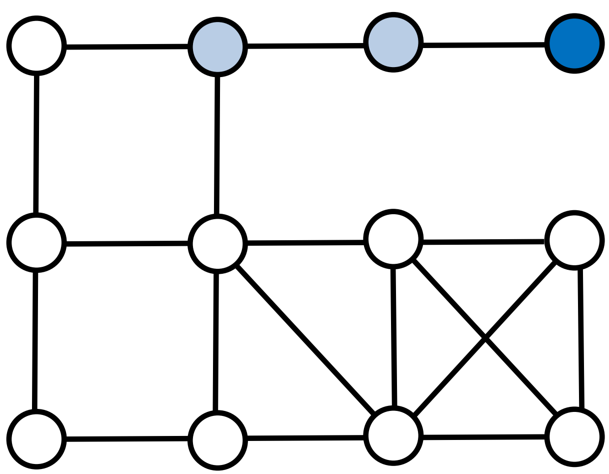
Heuristics

Neighborhood scaled closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the *ratio* of the resulting closure to the number of vertices added is maximized



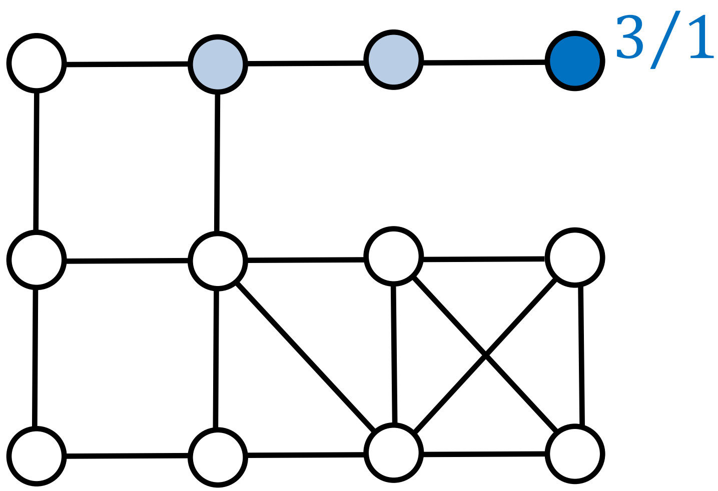
Heuristics

Neighborhood scaled closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the *ratio* of the resulting closure to the number of vertices added is maximized



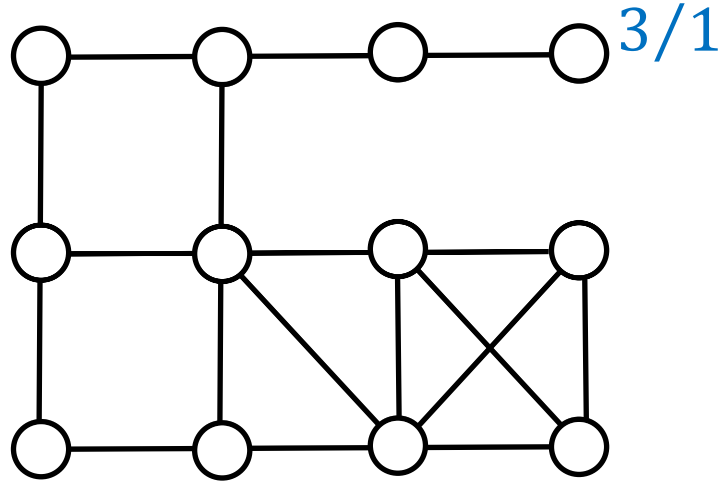
Heuristics

Neighborhood scaled closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the *ratio* of the resulting closure to the number of vertices added is maximized



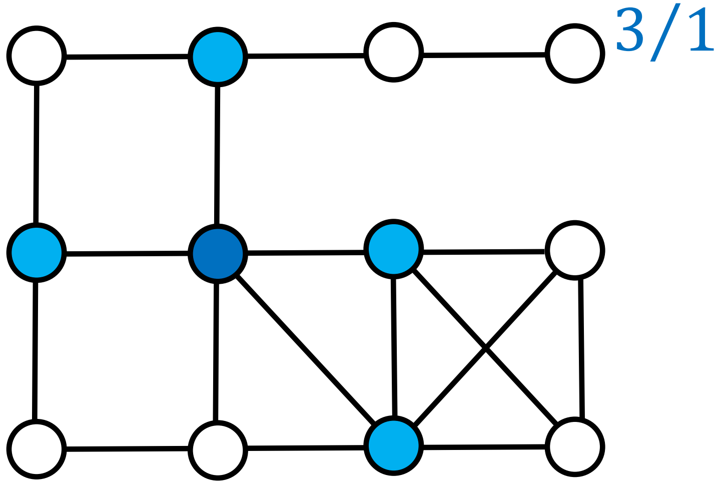
Heuristics

Neighborhood scaled closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the *ratio* of the resulting closure to the number of vertices added is maximized



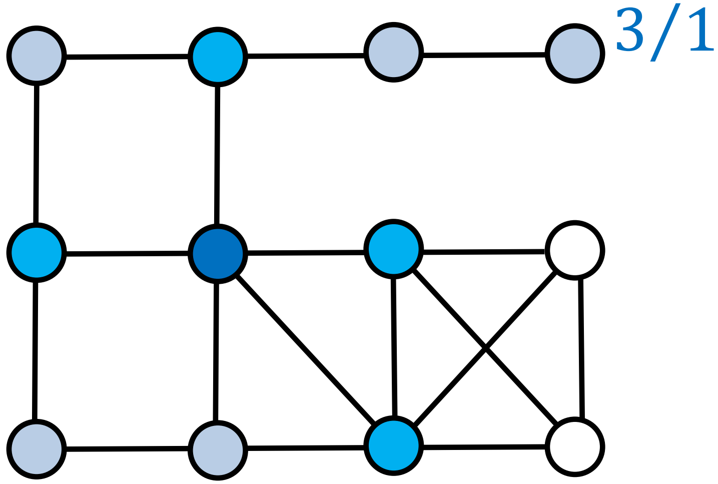
Heuristics

Neighborhood scaled closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the *ratio* of the resulting closure to the number of vertices added is maximized



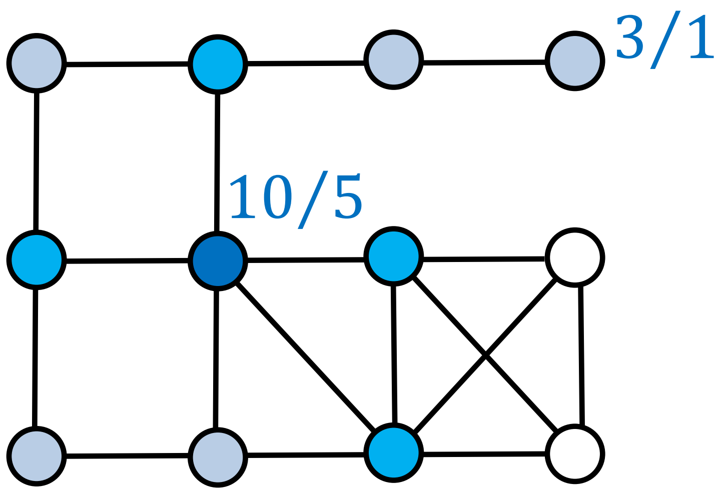
Heuristics

Neighborhood scaled closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the *ratio* of the resulting closure to the number of vertices added is maximized



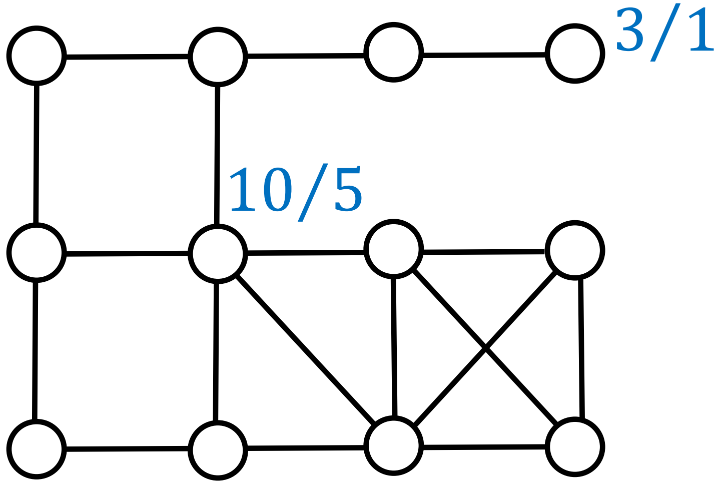
Heuristics

Neighborhood scaled closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the *ratio* of the resulting closure to the number of vertices added is maximized



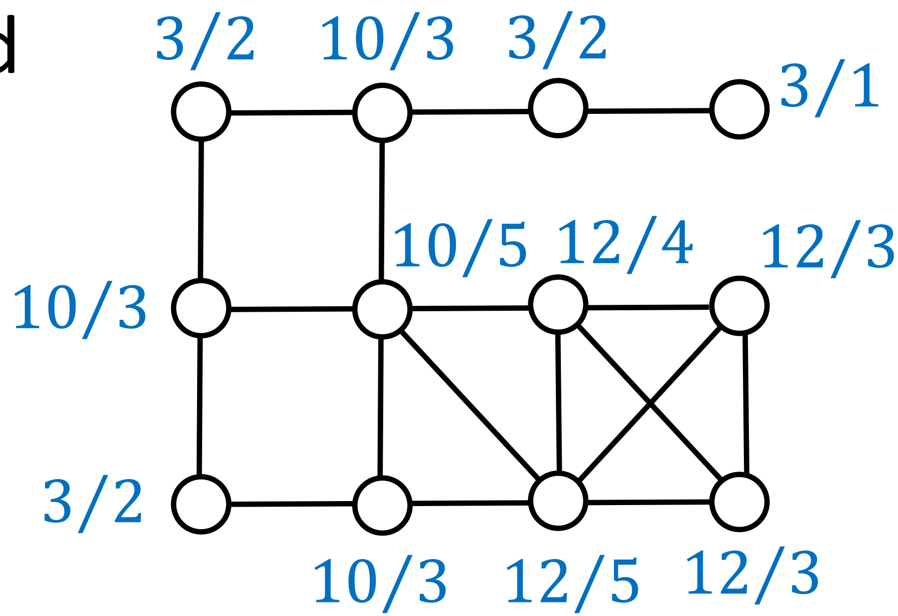
Heuristics

Neighborhood scaled closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the *ratio* of the resulting closure to the number of vertices added is maximized



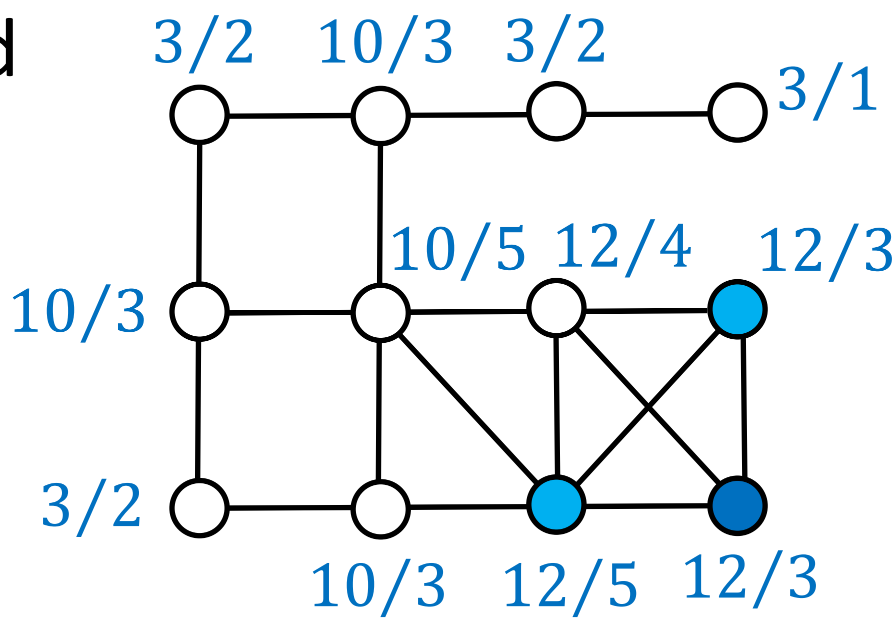
Heuristics

Neighborhood scaled closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the *ratio* of the resulting closure to the number of vertices added is maximized



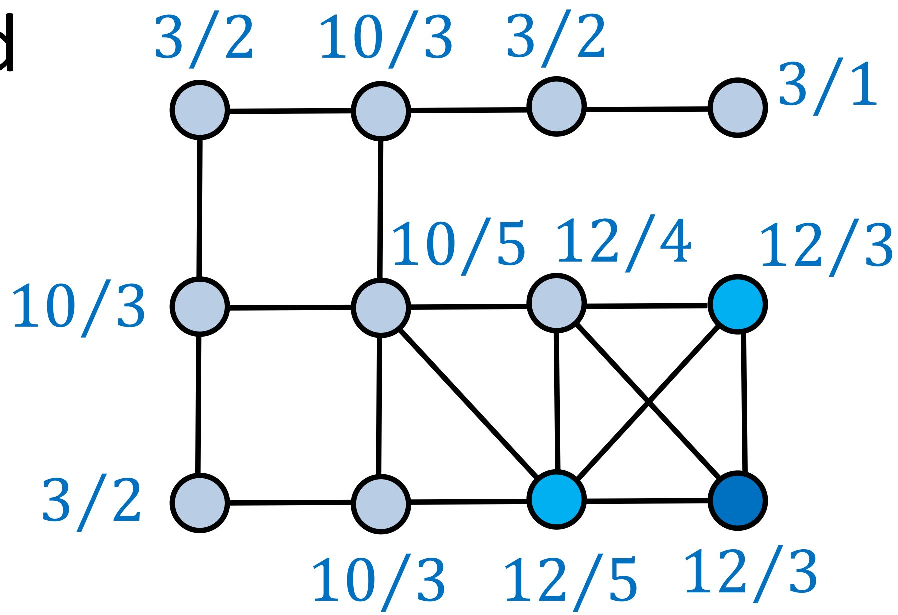
Heuristics

Neighborhood scaled closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the *ratio* of the resulting closure to the number of vertices added is maximized



Heuristics

Neighborhood scaled closure: add a single white vertex and all-but-one of its white neighbors to Z, so that the *ratio* of the resulting closure to the number of vertices added is maximized



Heuristic results

			Single Node Largest Closure		Neighborhood Largest Closure		Neighborhood Scaled Closure		
G	$\|V\|$	$Z(G)$	Z_h	time	Z_h	time	Z_h	time	
IEEE 14	14	4	**4**	0.000	5	0.000	5	0.000	✓
IEEE 24	24	6	7	0.001	**6**	0.000	**6**	0.000	✓
IEEE 30	30	7	**7**	0.000	9	0.000	**7**	0.000	✓
IEEE 39	39	7	9	0.003	**8**	0.001	**8**	0.001	✓
IEEE 57	57	9	**10**	0.007	**10**	0.002	11	0.003	✓
RTS 96	73	15	17	0.019	17	0.008	**16**	0.006	✓
IEEE 118	118	26	**27**	0.048	32	0.017	28	0.029	✓
karate	34	13	**13**	0.003	14	0.001	14	0.002	✓
chesapeake	39	14	**15**	0.007	**15**	0.002	**15**	0.002	✓
dolphins	62	14	**17**	0.015	19	0.003	**17**	0.004	✗
lesmis	77	40	**41**	0.040	**41**	0.008	**41**	0.026	✓

Heuristic results

G	$\|V\|$	$Z(G)$	Single Node Largest Closure Z_h	Single Node Largest Closure time	Neighborhood Largest Closure Z_h	Neighborhood Largest Closure time	Neighborhood Scaled Closure Z_h	Neighborhood Scaled Closure time	
WS(10,.3)	20	12.0	**12.2**	0.000	**12.2**	0.000	**12.2**	0.000	✓
	30	15.4	16.2	0.002	16.8	0.000	**16.0**	0.000	✓
	40	18.0	18.4	0.004	18.4	0.001	**18.2**	0.001	✓
	50	21.8	**22.0**	0.009	24.0	0.002	23.6	0.002	✓
	60	24.6	**24.8**	0.015	25.6	0.003	25.8	0.003	✓
	70	27.4	28.8	0.025	**28.4**	0.005	**28.4**	0.004	✓
	80	31.2	**32.6**	0.036	33.8	0.008	33.2	0.007	✗
WS(5,.3)	10	4.4	4.8	0.000	**4.6**	0.000	**4.6**	0.000	✓
	20	6.2	**6.2**	0.000	6.4	0.000	6.4	0.000	✓
	30	7.0	8.0	0.001	**7.4**	0.000	7.6	0.000	✓
	40	9.4	10.2	0.002	10.6	0.000	**9.6**	0.000	✓
	50	10.8	**11.6**	0.005	12.2	0.002	**11.6**	0.002	✓
	60	11.6	13.6	0.008	14.0	0.002	**13.2**	0.002	✗
	70	14.0	15.0	0.016	16.2	0.003	**14.8**	0.004	✓
	80	14.8	**16.8**	0.015	18.2	0.005	17.2	0.006	✗

Heuristic results

			Single Node Largest Closure		Neighborhood Largest Closure		Neighborhood Scaled Closure		
G	$\lvert V \rvert$	$Z(G)$	Z_h	time	Z_h	time	Z_h	time	
Cubic	10	3.8	**3.8**	0.000	**3.8**	0.000	**3.8**	0.000	✓
	20	5.2	5.4	0.000	**5.2**	0.000	**5.2**	0.000	✓
	30	6.6	7.4	0.001	7.2	0.000	**7.0**	0.000	✓
	40	8.8	9.8	0.003	9.6	0.001	**9.2**	0.001	✓
	50	9.2	10.4	0.005	**9.4**	0.001	9.6	0.001	✓
	60	11.4	12.4	0.008	12.0	0.003	**11.8**	0.003	✓
	70	12.0	**12.6**	0.008	13.2	0.004	12.8	0.004	✓

- Compared different methods for computing $Z(G)$
- Each has its pros and cons

- Compared different methods for computing $Z(G)$
- Each has its pros and cons
- Presented heuristics for $Z(G)$
  - Can speed up exact models
  - Very fast and usually pretty accurate
  - Can be adapted to other graph infection problems

- Compared different methods for computing $Z(G)$
- Each has its pros and cons
- Presented heuristics for $Z(G)$
  - Can speed up exact models
  - Very fast and usually pretty accurate
  - Can be adapted to other graph infection problems
- Future work: develop timestep-based SAT model, derive performance guarantees for heuristics

- Compared different methods for computing $Z(G)$
- Each has its pros and cons
- Presented heuristics for $Z(G)$
  - Can speed up exact models
  - Very fast and usually pretty accurate
  - Can be adapted to other graph infection problems
- Future work: develop timestep-based SAT model, derive performance guarantees for heuristics

- B. Brimkov, I. V. Hicks, D. J. Mikesell. Improved computational approaches and heuristics for zero forcing *INFORMS Journal on Computing* (2020).
- B. Brimkov, C.C. Fast, I.V. Hicks. Computational approaches for zero forcing and related problems. *European Journal of Operational Research* (2019).

- Compared different methods for computing $Z(G)$
- Each has its pros and cons
- Presented heuristics for $Z(G)$
  - Can speed up exact models
  - Very fast and usually pretty accurate
  - Can be adapted to other graph infection problems
- Future work: develop timestep-based SAT model, derive performance guarantees for heuristics

- B. Brimkov, I. V. Hicks, D. J. Mikesell. Improved computational approaches and heuristics for zero forcing *INFORMS Journal on Computing* (2020).
- B. Brimkov, C.C. Fast, I.V. Hicks. Computational approaches for zero forcing and related problems. *European Journal of Operational Research* (2019).

THANK YOU