The Maximum Independent Set Problem is Easy

Darren Strash

Shonan Meeting 144 | March 5, 2019

The Maximum Independent Set Problem is Easy

(Except When it Isn't)

Darren Strash

Shonan Meeting 144 | March 5, 2019

Applications

Computer vision:

→ Image segmentation

†


† http://www.ntu.edu.sg/home/asjfcai/Benchmark_Website/benchmark_index.html

Applications

Computer vision:

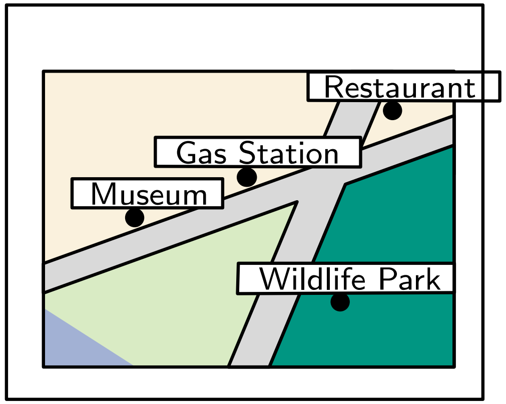
→ Image segmentation

†

† http://www.ntu.edu.sg/home/asjfcai/Benchmark_Website/benchmark_index.html

Map labeling:

→ Maximize nonoverlapping labels



Applications

Computer vision:

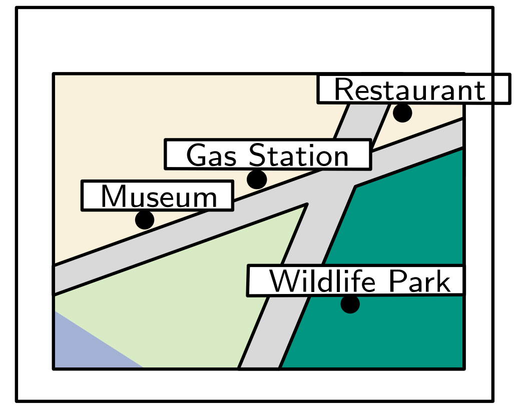
→ Image segmentation

†

† http://www.ntu.edu.sg/home/asjfcai/Benchmark_Website/benchmark_index.html

Map labeling:

→ Maximize nonoverlapping labels



Tracking submarines:

*

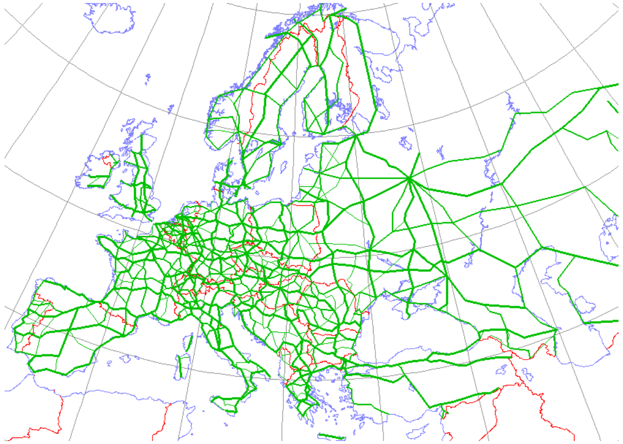
→ Coordinate information from multiple sensors

Large real-world networks

Graphs with millions/billions of nodes and "structure"

$\rightarrow$ social networks, web-crawl graphs, co-citation networks

$\rightarrow$ sparse, many low-degree vertices

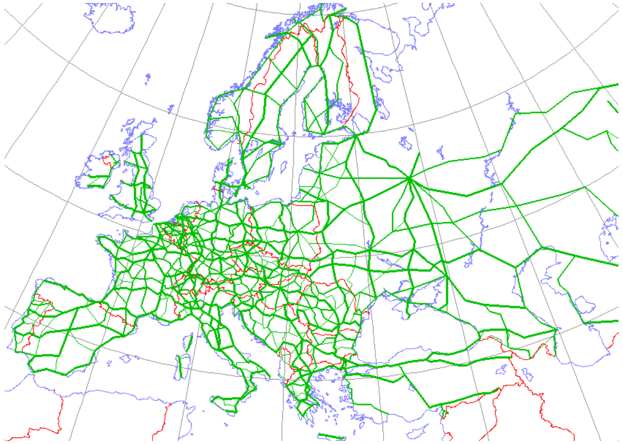


Large real-world networks

Graphs with millions/billions of nodes and "structure"

$\rightarrow$ social networks, web-crawl graphs, co-citation networks

$\rightarrow$ sparse, many low-degree vertices



Branch-and bound is limited to hundreds (maybe thousands) of vertices

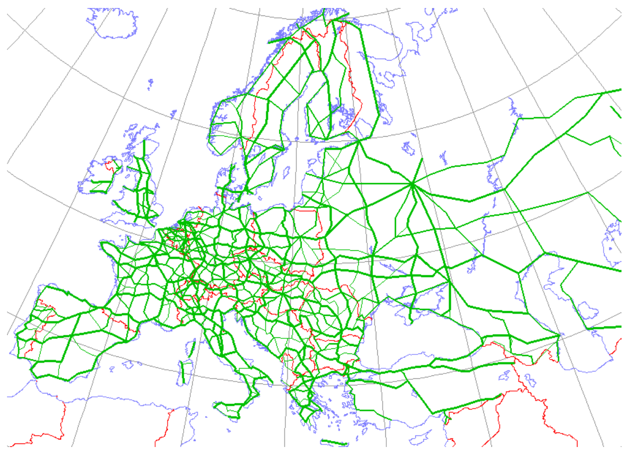
$\rightarrow$ graph C4000.5 solved with 1 year of computation!

Large real-world networks

Graphs with millions/billions of nodes and "structure"

→ social networks, web-crawl graphs, co-citation networks

→ sparse, many low-degree vertices



Branch-and bound is limited to hundreds (maybe thousands) of vertices

→ graph C4000.5 solved with 1 year of computation!

Sparse graphs should be worse...

Large real-world networks

$\alpha(G)$ is (roughly) linear for sparse graphs...

$\rightarrow$ linear search depth is infeasible for branch and bound

Large real-world networks

$\alpha(G)$ is (roughly) linear for sparse graphs...

$\rightarrow$ linear search depth is infeasible for branch and bound

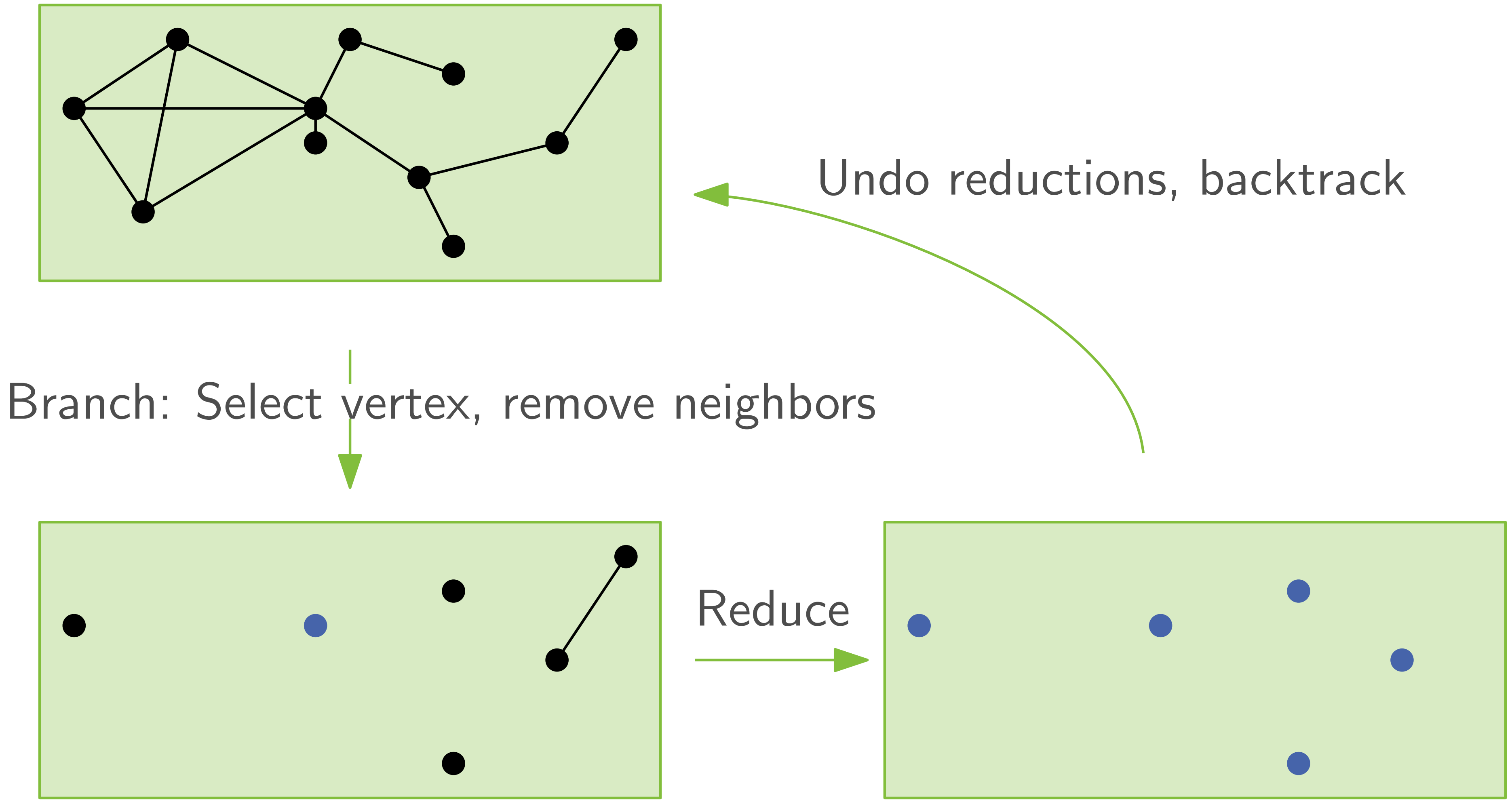
...enter inexact algorithms!

Graph		EvoMIS		
Name	n	Avg.	Max.	Min.
enron	69 244	62 811	**62 811**	62 811
gowalla	196 591	112 369	**112 369**	112 369
citation	268 495	150 380	**150 380**	150 380
cnr-2000*	325 557	229 981	**229 991**	229 976
google	356 648	174 072	**174 072**	174 072
coPapers	434 102	47 996	**47 996**	47 996
skitter*	554 930	328 519	328 520	328 519
amazon	735 323	309 774	309 778	309 769
in-2004*	1 382 908	896 581	**896 585**	896 580

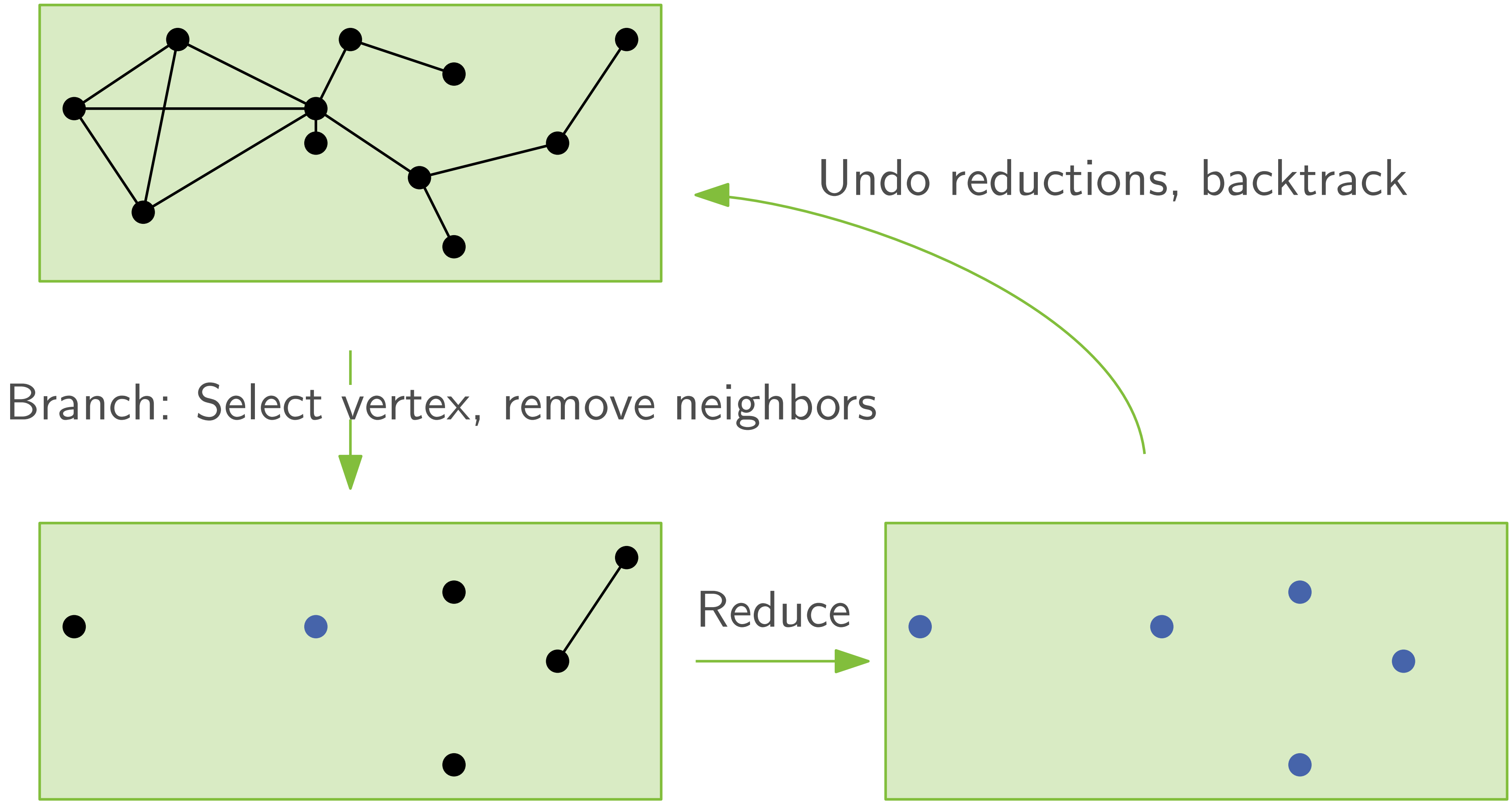
Branch and Reduce

[Akiba and Iwata, 2016]

Branch-and-reduce algorithms



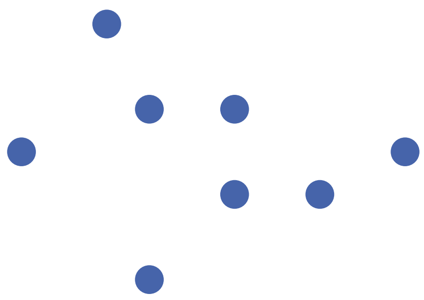
Branch-and-reduce algorithms



$\rightarrow$ Effective in theory: $O^*(1.1996^n)$ [Xiao and Nagamochi, 2017]

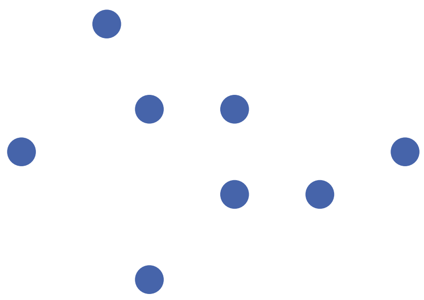
Reduction rules

Degree 0

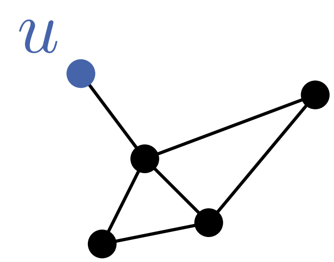


Reduction rules

Degree 0

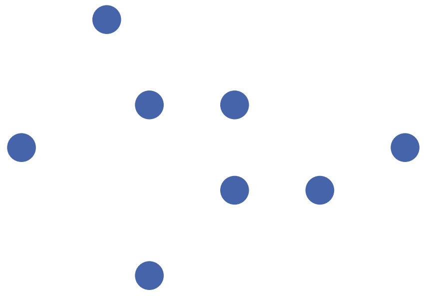


Degree 1

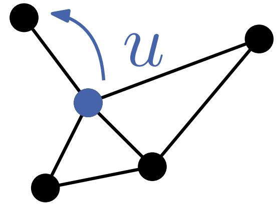


Reduction rules

Degree 0

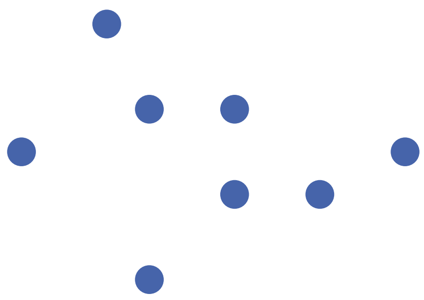


Degree 1

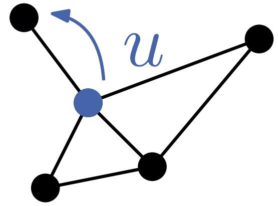


Reduction rules

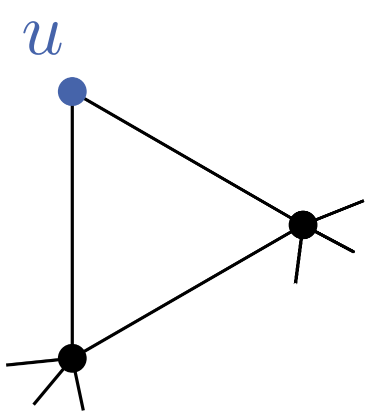
Degree 0



Degree 1

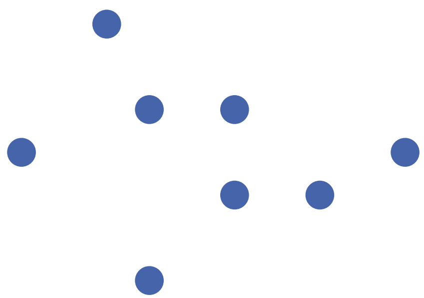


Degree 2

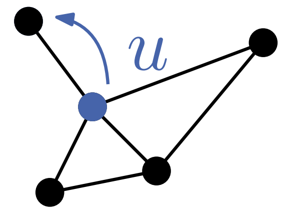


Reduction rules

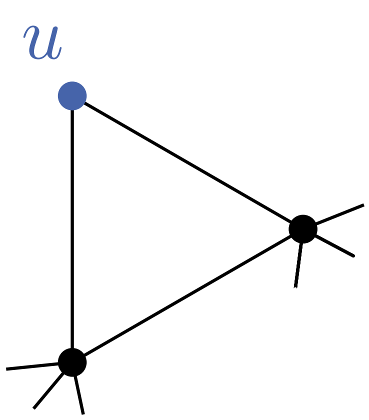
Degree 0



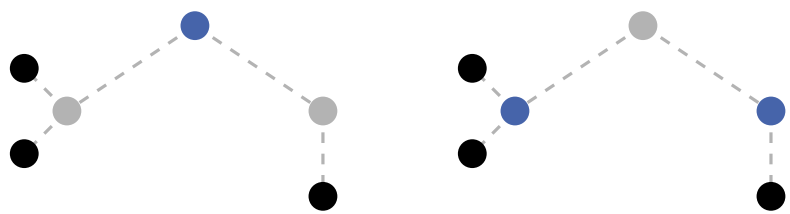
Degree 1



Degree 2

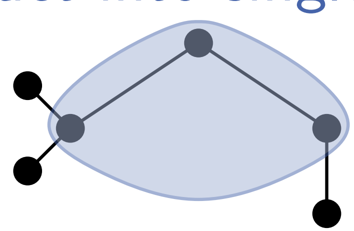


vertex folding



↓

Contract into single vertex

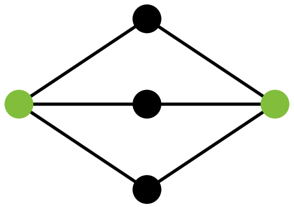


And more... [Akiba and Iwata, 2016]

Reductions

- LP-relaxation

  → Maximize $\sum x_v$ where $x_u + x_v \leq 1$. If $x_v = 1$, then **in some MIS**.
- Unconfined
  → **Some MIS** exists without "unconfined" vertices
- Twin
  → Generalization of vertex folding



- Diamond, alternative, . . .

And more...

[Akiba and Iwata, 2016]

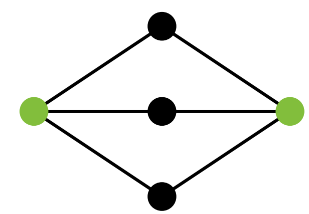
Reductions

- LP-relaxation

  → Maximize $\sum x_v$ where $x_u + x_v \leq 1$. If $x_v = 1$, then **in some MIS**.
- Unconfined

  → **Some MIS** exists without "unconfined" vertices
- Twin

  → Generalization of vertex folding



- Diamond, alternative, ...

Other techniques

- Packing constraints

  → Maintain constraints that update throughout recursion
- Branching rules, vertex ordering, ...

Works well!

On LAW, SNAP, KONECT graphs...

- Solves in less than 1 second:
  - Citation networks (≈200,000 vertices)
  - Web crawl graphs (≈500,000 vertices)
  - Social networks (≈3,000,000 vertices)

Works well!

On LAW, SNAP, KONECT graphs...

- Solves in less than 1 second:
  - Citation networks (≈200,000 vertices)
  - Web crawl graphs (≈500,000 vertices)
  - Social networks (≈3,000,000 vertices)

- Some take much longer:
  - as-skitter (big) (1,170,580 vertices) **48 min**
  - web-Stanford (163,390 vertices) **13 hours**
  - Many networks remain **unsolved**

Works well!

On LAW, SNAP, KONECT graphs...

- Solves in less than 1 second: $\rightarrow$ why?
  - Citation networks ($\approx$200,000 vertices)
  - Web crawl graphs ($\approx$500,000 vertices)
  - Social networks ($\approx$3,000,000 vertices)

- Some take much longer: $\rightarrow$ why?
  - as-skitter (big) (1,170,580 vertices) **48 min**
  - web-Stanford (163,390 vertices) **13 hours**
  - Many networks remain **unsolved**

Works well!

On LAW, SNAP, KONECT graphs...

- Solves in less than 1 second: → why?
  - Citation networks (≈200,000 vertices)
  - Web crawl graphs (≈500,000 vertices)
  - Social networks (≈3,000,000 vertices)

- Some take much longer: → why?
  - as-skitter (big) (1,170,580 vertices) **48 min**
  - web-Stanford (163,390 vertices) **13 hours**
  - Many networks remain **unsolved**

  → what can be done about it?

Works well!

On LAW, SNAP, KONECT graphs...

- Solves in less than 1 second: → why?
  - Citation networks (≈200,000 vertices)
  - Web crawl graphs (≈500,000 vertices)
  - Social networks (≈3,000,000 vertices)

reductions are powerful

- Some take much longer: → why?
  - as-skitter (big) (1,170,580 vertices) **48 min**
  - web-Stanford (163,390 vertices) **13 hours**
  - Many networks remain **unsolved**

→ what can be done about it?

Remove redundant computation

Combine reductions and heuristic search

The power of simple reductions

[Strash, 2016]

An explanation

- Solves in less than 1 second: $\rightarrow$ why?
  - Citation networks (≈200,000 vertices)
  - Web crawl graphs (≈500,000 vertices)
  - Social networks (≈3,000,000 vertices)

An explanation

- Solves in less than 1 second: → why?
  - Citation networks (≈200,000 vertices)
  - Web crawl graphs (≈500,000 vertices)
  - Social networks (≈3,000,000 vertices)

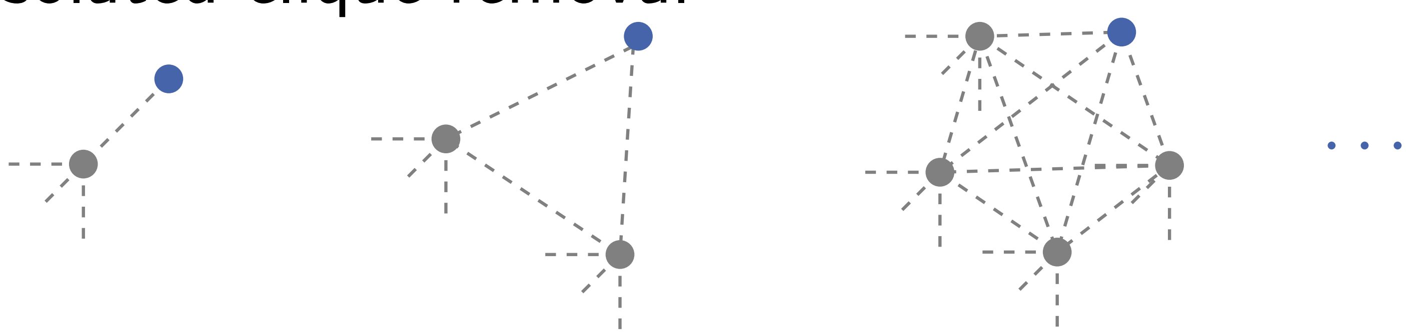
→ After reductions, nearly all graphs are empty.

An explanation

- Solves in less than 1 second: $\rightarrow$ why?
  - Citation networks ($\approx$200,000 vertices)
  - Web crawl graphs ($\approx$500,000 vertices)
  - Social networks ($\approx$3,000,000 vertices)

$\rightarrow$ After reductions, nearly all graphs are empty.

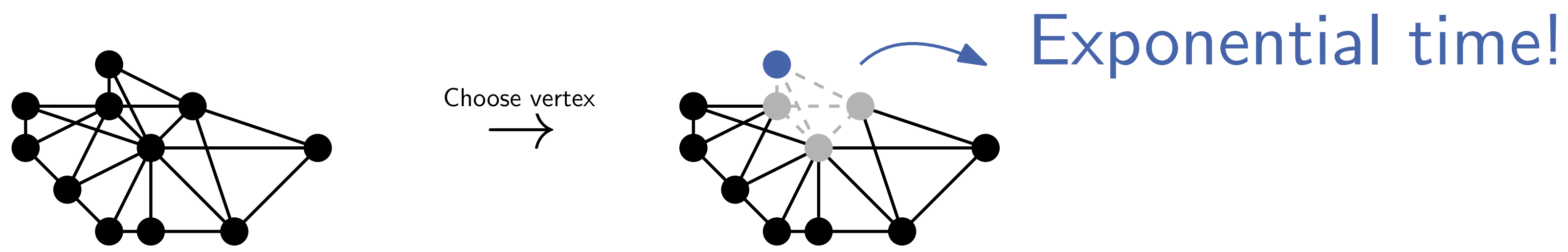
- 80% instances solved with two reductions (< 1 sec)
  - Vertex folding
  - Isolated clique removal



Combining reductions and inexact algorithms

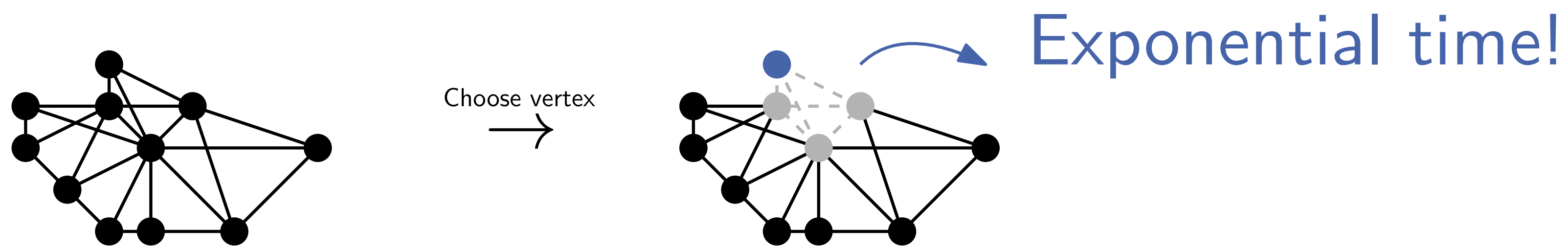
Heuristic: Guess "likely candidates" [Lamm et al. 2016]

- Branch-and-reduce selects **one solution vertex** at a time
  $\rightarrow$ Limits the number of reductions in next recursion call



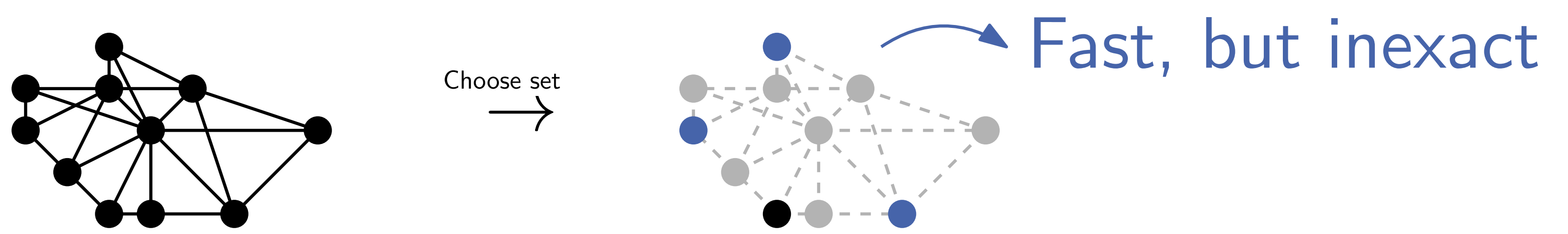
Heuristic: Guess "likely candidates" [Lamm et al. 2016]

- Branch-and-reduce selects **one solution vertex** at a time
  → Limits the number of reductions in next recursion call



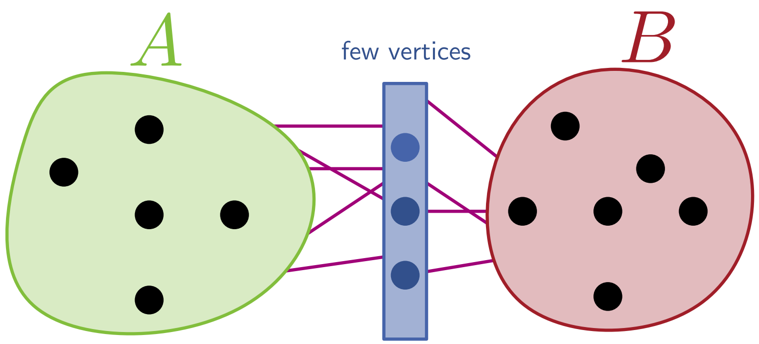
- Can we guess **many vertices** that are likely in an MIS?

  → Remove and continue applying reductions



Evolutionary algorithm [EvoMIS] [Lamm et al. 2015]

- Start with an initial independent set I
- Swap whole blocks of independent set nodes using **separators** and **graph partitioning**

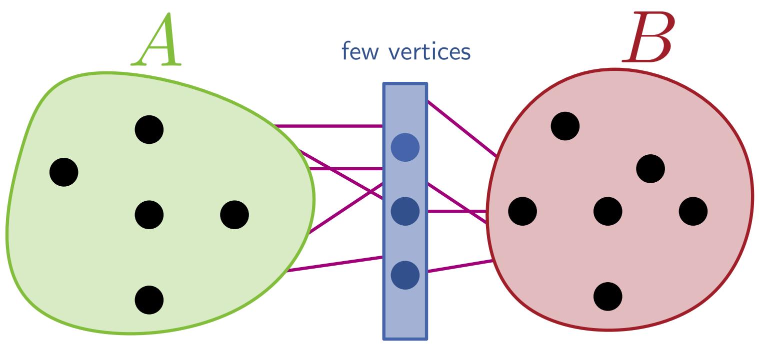


Update solutions to A and B with **local search**

$\rightarrow$ next generation

Evolutionary algorithm [EvoMIS] [Lamm et al. 2015]

- Start with an initial independent set I
- Swap whole blocks of independent set nodes using **separators** and **graph partitioning**



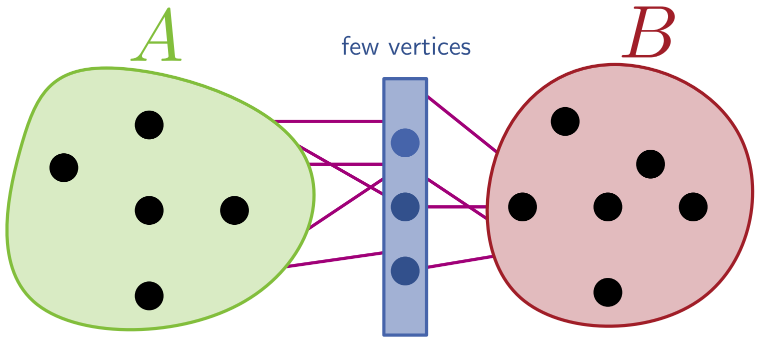
Update solutions to A and B with **local search**

$\rightarrow$ next generation

- Finds large independent sets in large sparse networks.

Evolutionary algorithm [EvoMIS] [Lamm et al. 2015]

- Start with an initial independent set I
- Swap whole blocks of independent set nodes using **separators** and **graph partitioning**



Update solutions to A and B with **local search**

$\rightarrow$ next generation

- Finds large independent sets in large sparse networks.

Idea: Select low-degree vertices from "fittest" independent set.

ReduMIS: Near-optimal on “difficult networks”

- Finds exact MIS faster, when exact algorithm is slow:
  - as-skitter (big) **48 min** $\rightarrow$ **21 min**
  - web-Stanford **13 hours** $\rightarrow$ **5 min**
  - bcsstk30 **8.6 hours** $\rightarrow$ **2.4 sec**
  - brack2 **13 min** $\rightarrow$ **9.4 sec**
  - col **2 hours** $\rightarrow$ **28 sec**

ReduMIS: Near-optimal on “difficult networks”

- Finds exact MIS faster, when exact algorithm is slow:
  - as-skitter (big) **48 min → 21 min**
  - web-Stanford **13 hours → 5 min**
  - bcsstk30 **8.6 hours → 2.4 sec**
  - brack2 **13 min → 9.4 sec**
  - col **2 hours → 28 sec**
- Finds exact MIS, for large networks with known MIS size

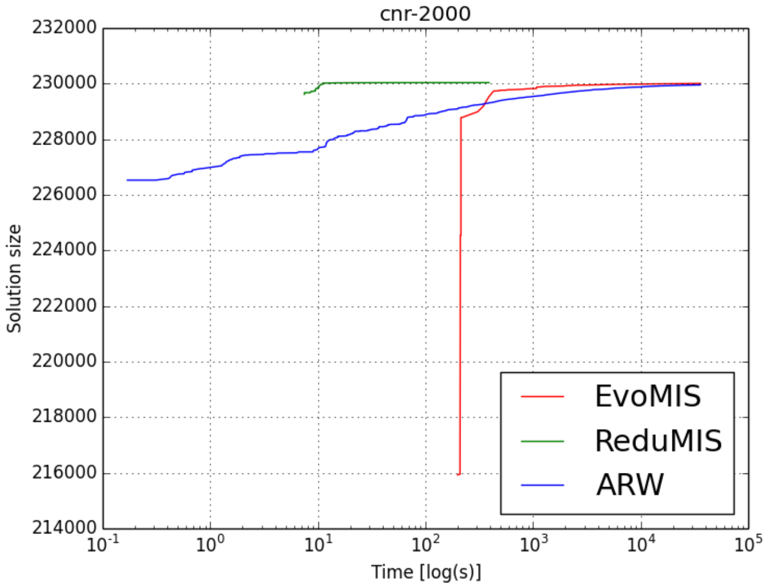
ReduMIS: Near-optimal on "difficult networks"

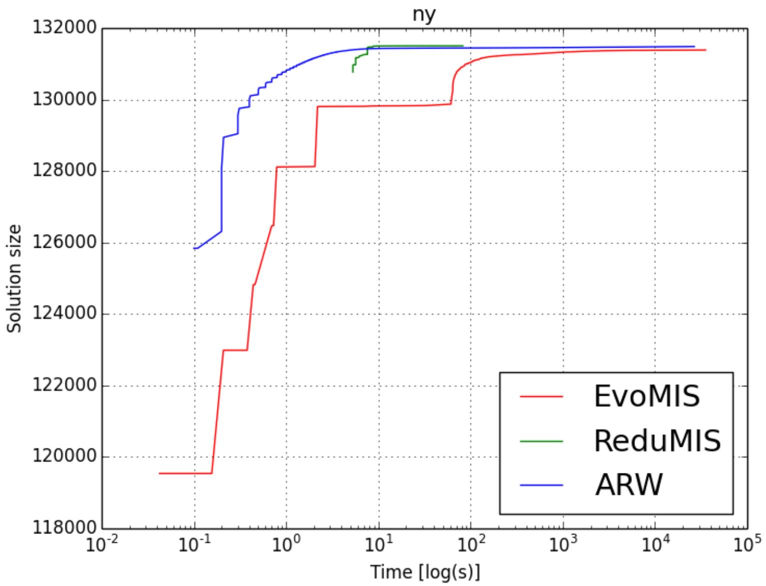
- Finds exact MIS faster, when exact algorithm is slow:
  - as-skitter (big) **48 min → 21 min**
  - web-Stanford **13 hours → 5 min**
  - bcsstk30 **8.6 hours → 2.4 sec**
  - brack2 **13 min → 9.4 sec**
  - col **2 hours → 28 sec**
- Finds exact MIS, for large networks with known MIS size
- Consistently finds same value for **large** graphs with unknown MIS size
  - cnr-2000 → **230,036**
  - skitter → **328,626**
  - amazon → **309,794**
  - ny → **131,502**

ReduMIS: Finds larger solutions faster

- Consistently finds larger solutions on social, Web, and road networks

  $\rightarrow$ averaged over 5 runs.

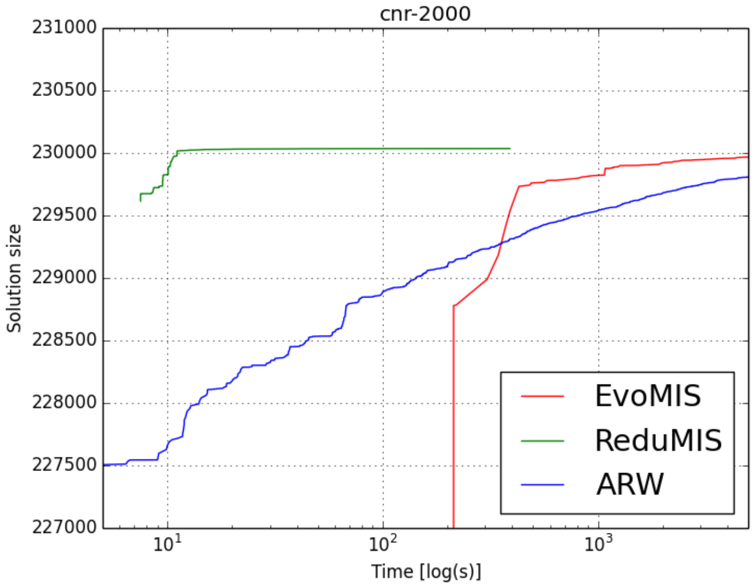


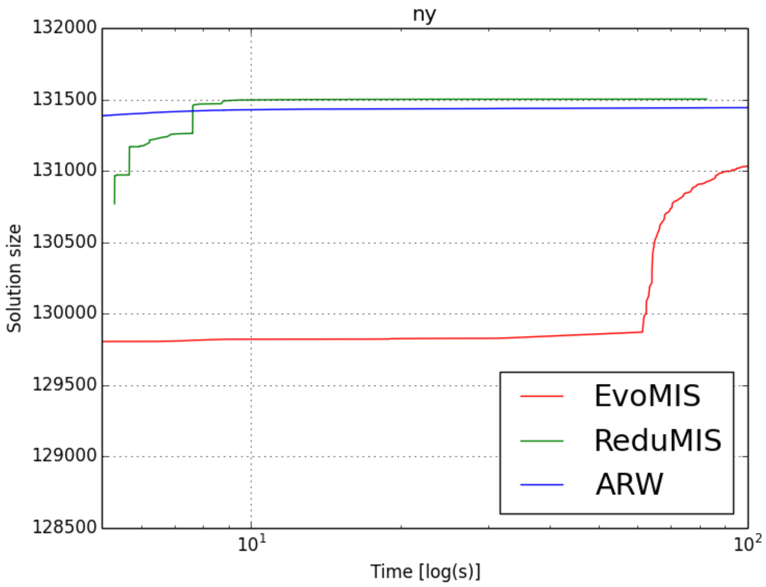


ReduMIS: Finds larger solutions faster

- Consistently finds larger solutions on social, Web, and road networks

  → averaged over 5 runs.

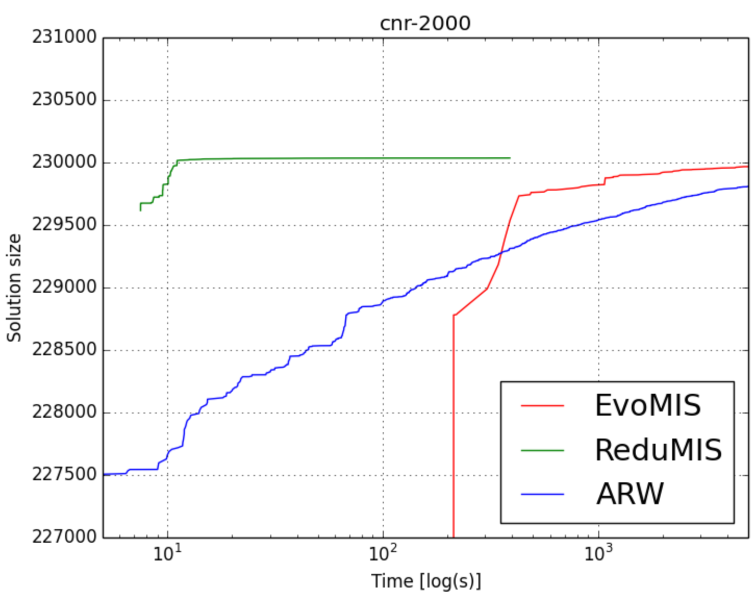


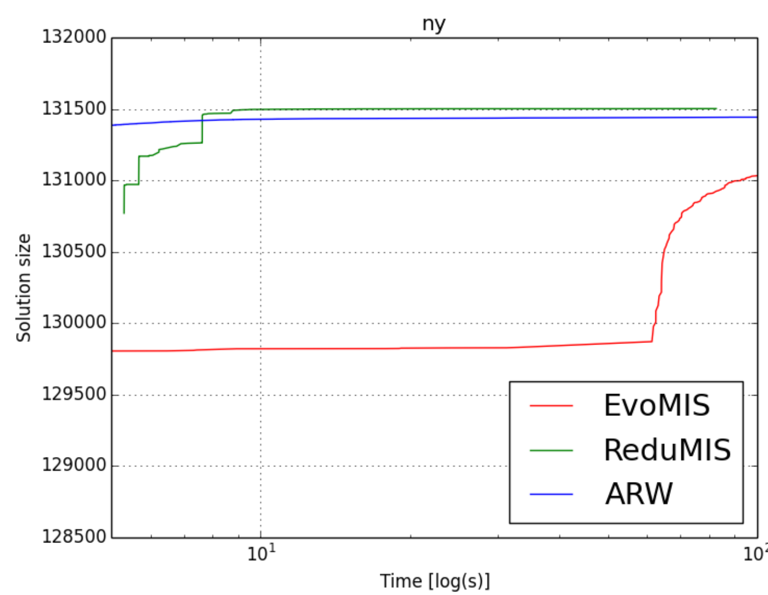


ReduMIS: Finds larger solutions faster

- Consistently finds larger solutions on social, Web, and road networks

  → averaged over 5 runs.

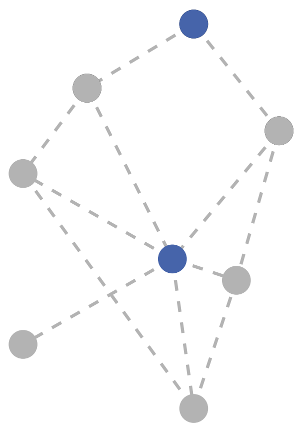




- Consistent, even as we scale to graphs on **10M to 100M nodes**
- However, finds worse solutions for huge meshes

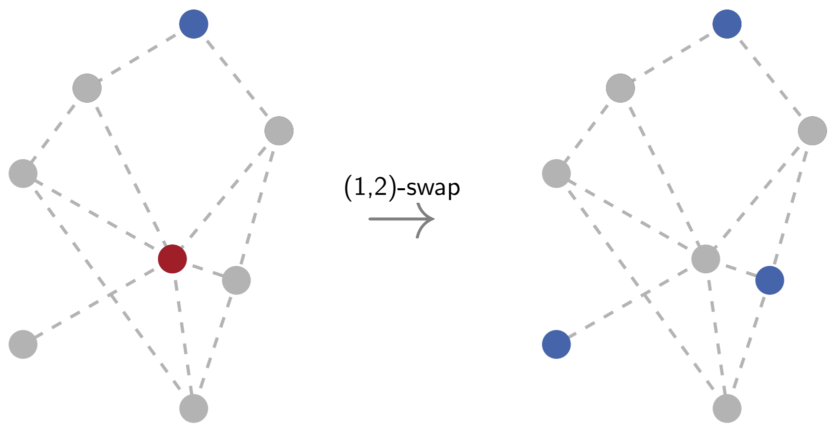
Iterated Local Search [ARW] [Andrade et al. 2012]

- Start with some maximal independent set
- (1,2)-swaps $\rightarrow$ remove one vertex, add two
- avoid swapping "recently" swapped vertices
- When not possible, perturb the solution



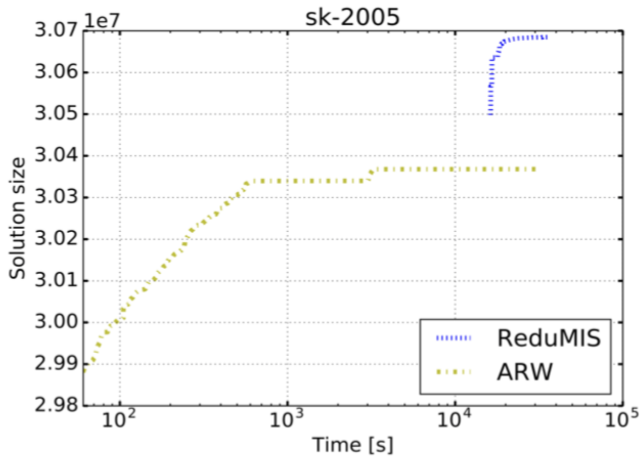
Iterated Local Search [ARW] [Andrade et al. 2012]

- Start with some maximal independent set
- (1,2)-swaps $\rightarrow$ remove one vertex, add two
- avoid swapping "recently" swapped vertices
- When not possible, perturb the solution



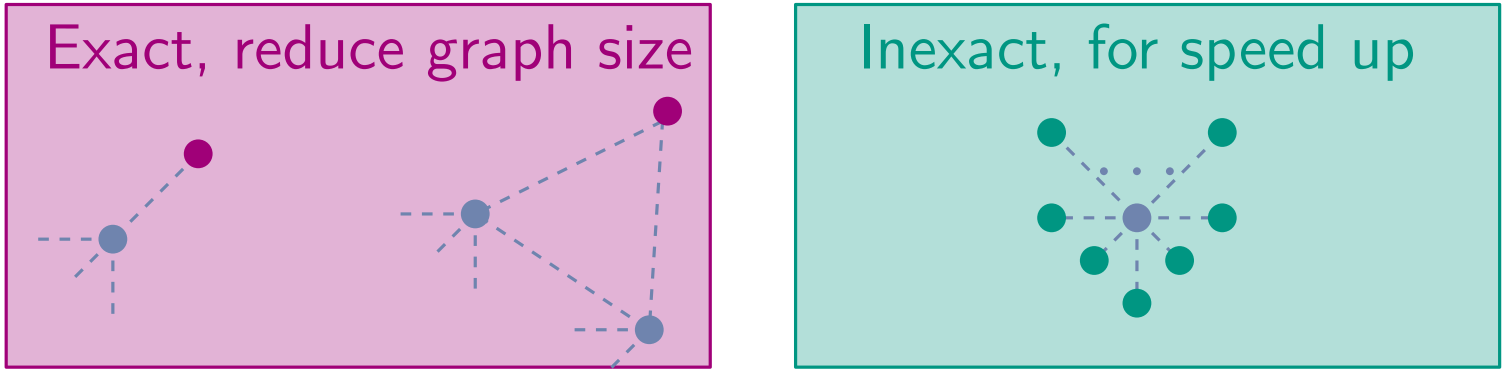
Accelerating Local Search [Dahlum et al. 2016]

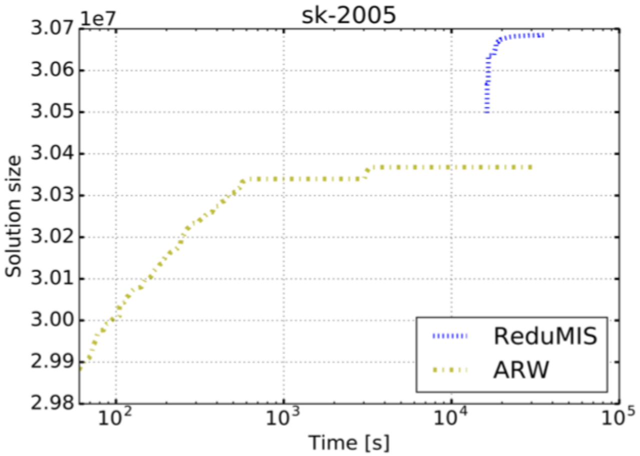
- **Problem:** Too much time to wait for high-quality solution.



Accelerating Local Search [Dahlum et al. 2016]

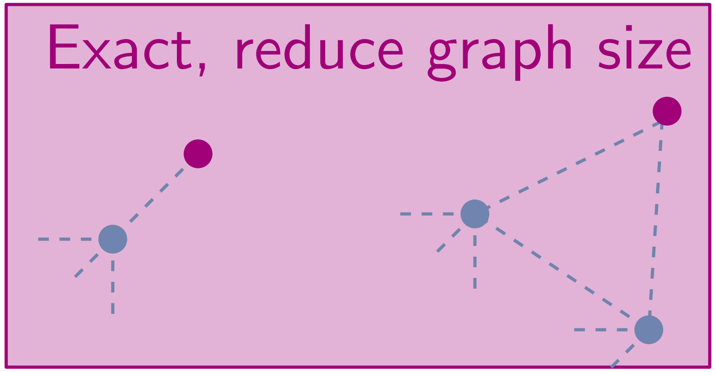
- **Problem:** Too much time to wait for high-quality solution.
- **Solution:** Speed up local search with online reductions, and removing high-degree vertices.

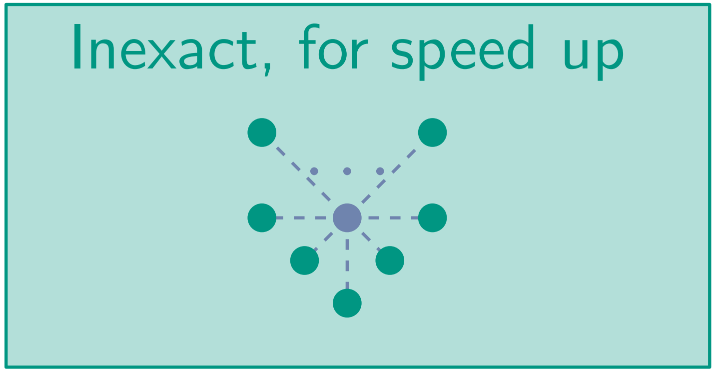


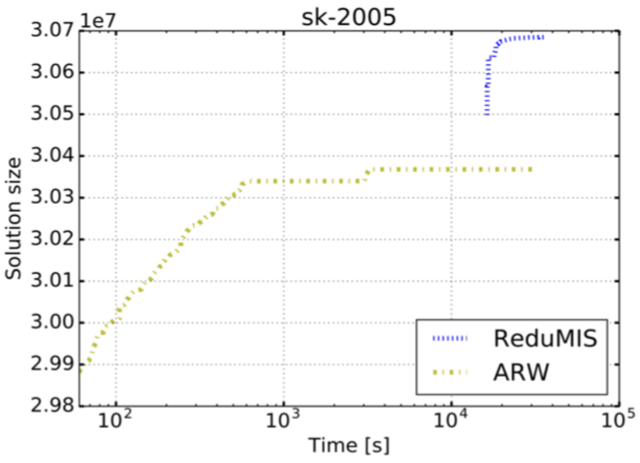


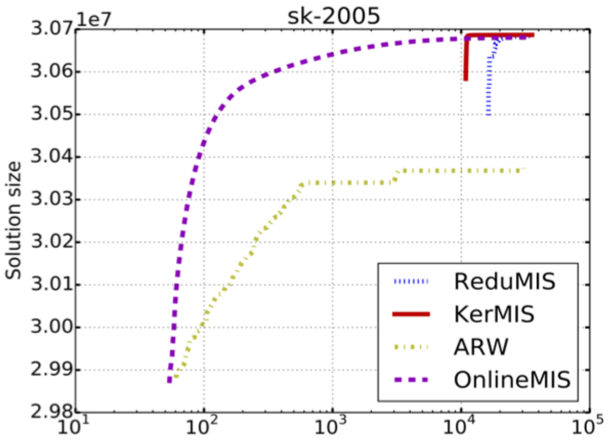
Accelerating Local Search [Dahlum et al. 2016]

- **Problem:** Too much time to wait for high-quality solution.
- **Solution:** Speed up local search with online reductions, and removing high-degree vertices.



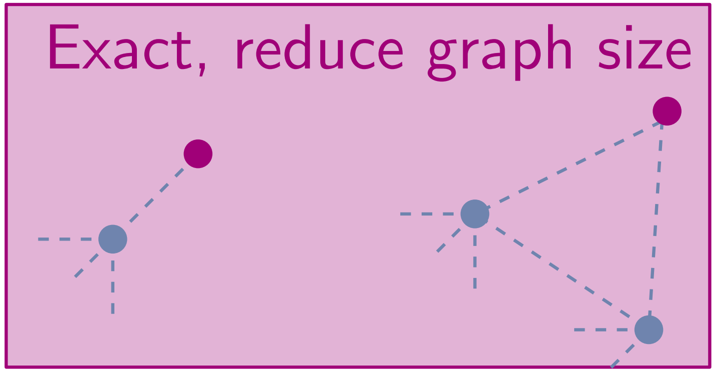


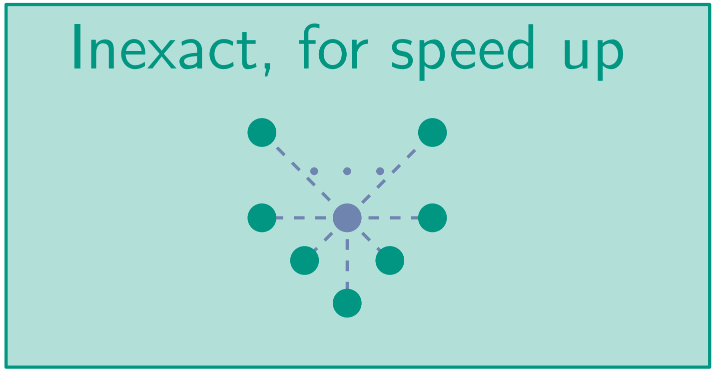


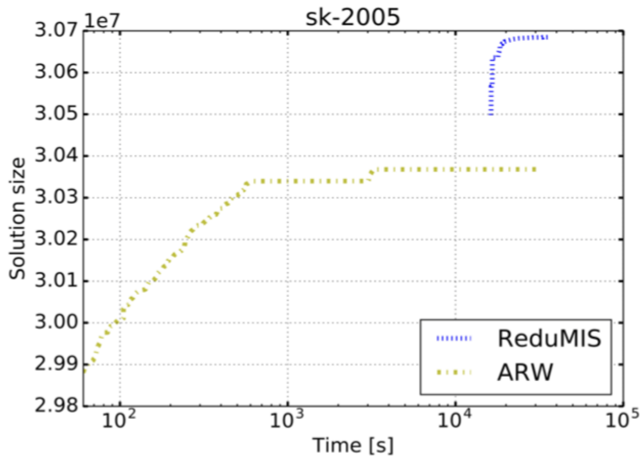


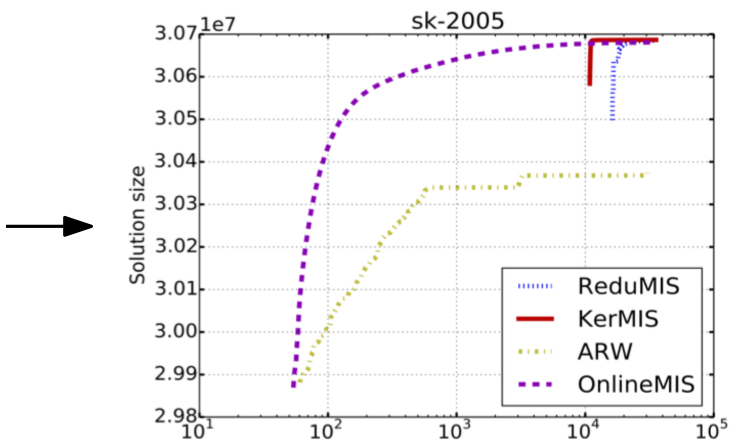
Accelerating Local Search [Dahlum et al. 2016]

- **Problem:** Too much time to wait for high-quality solution.
- **Solution:** Speed up local search with online reductions, and removing high-degree vertices.





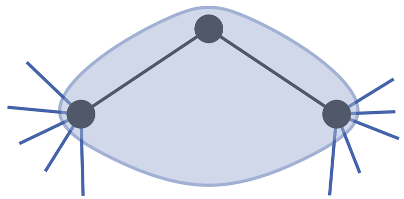




300x faster!

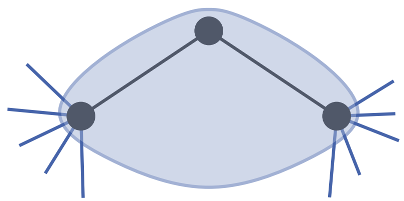
Linear-time reductions [Chang et al. 2017]

Problem: Vertex folding is slow with high-degree neighbors

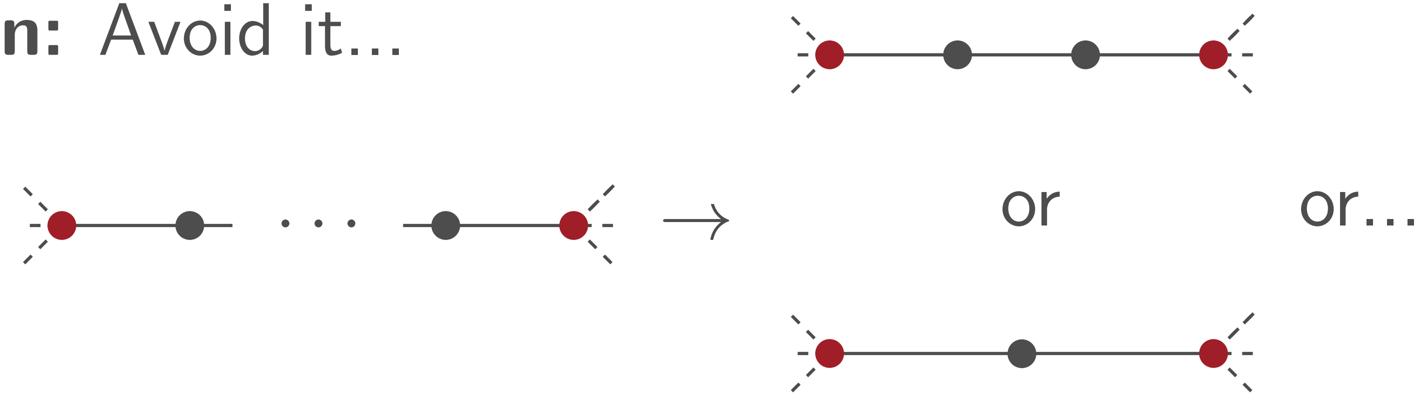


Linear-time reductions [Chang et al. 2017]

Problem: Vertex folding is slow with high-degree neighbors

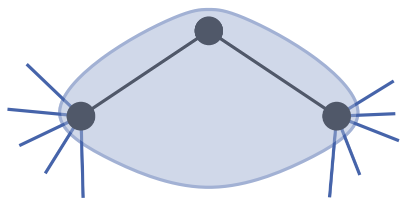


Solution: Avoid it...

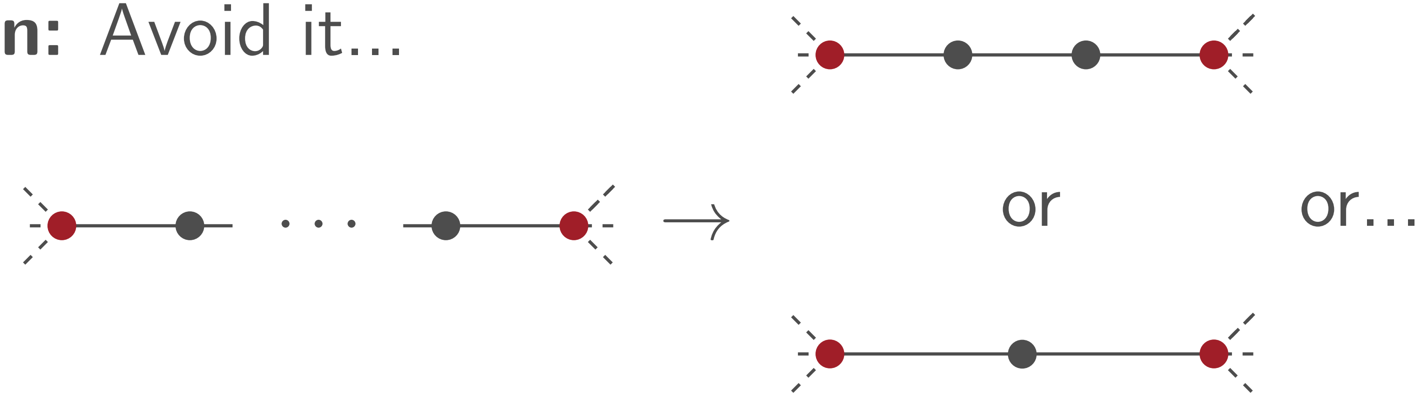


Linear-time reductions [Chang et al. 2017]

Problem: Vertex folding is slow with high-degree neighbors

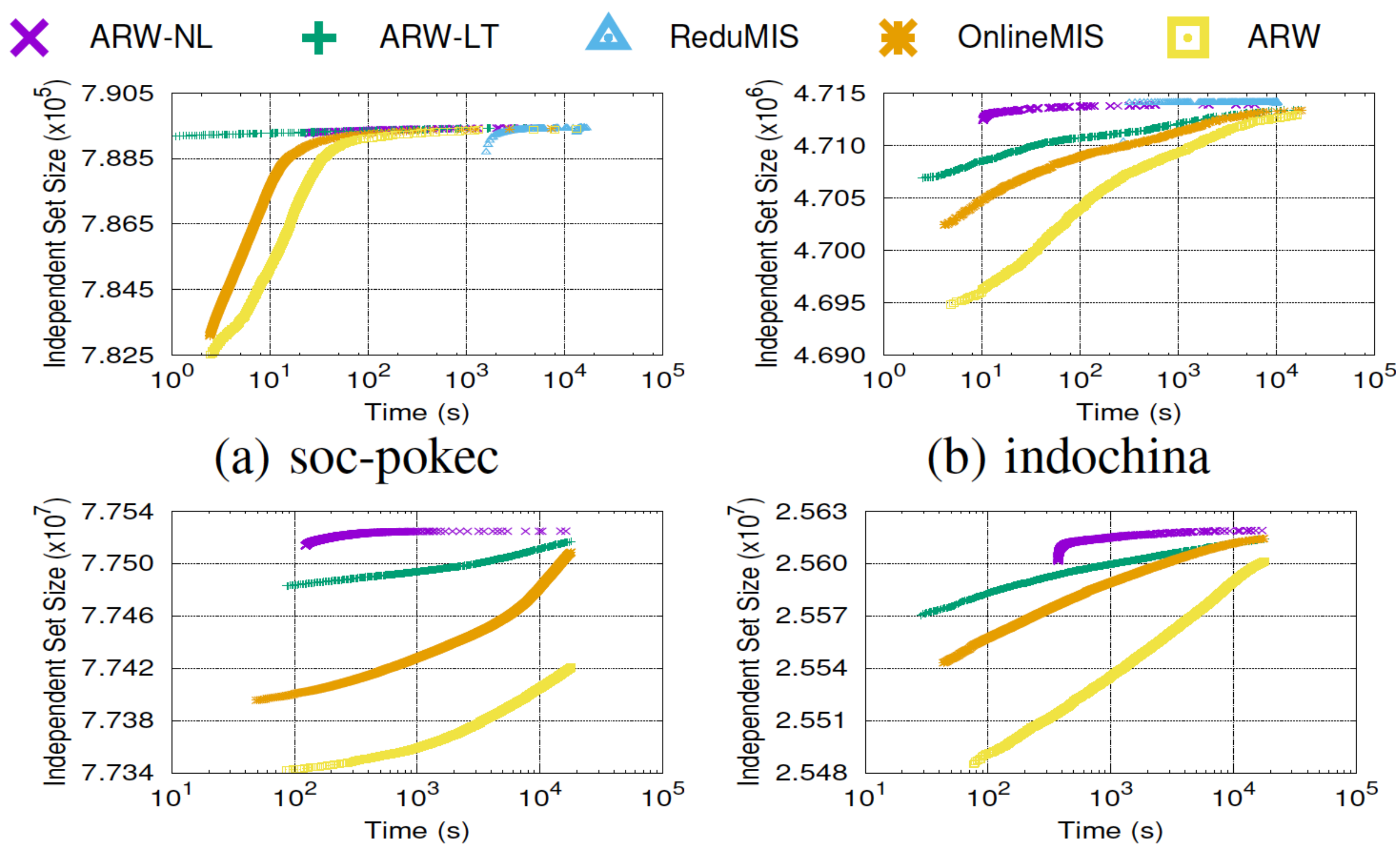


Solution: Avoid it...



Repeat: Add small degree vertex to solution + reduce

Linear-time reductions [Chang et al. 2017]



(a) soc-pokec (b) indochina

(c) webbase (d) it-2004

Scalable Reductions [Hespe et al. 2018]

Problem: Effective reductions are slow

Graph		LinearTime		NearLinear		VCSolver	
name	n	$\lvert\mathcal{K}\rvert$	time	$\lvert\mathcal{K}\rvert$	time	$\lvert\mathcal{K}\rvert$	time
uk-2002	19M	11.7M	1.5	4.0M	28.0	0.2M	336.9
arabic-2005	23M	15.6M	2.6	6.7M	246.1	0.6M	1 033.2
gsh-2015-tpd	31M	2.0M	11.6	1.2M	97.4	0.4M	372.3
uk-2005	39M	28.2M	2.5	5.9M	60.5	0.8M	541.4
it-2004	41M	27.1M	3.3	11.3M	1 544.6	1.6M	6 749.0
sk-2005	51M	*	*	*	*	3.2M	10 010.5
uk-2007-05	106M	*	*	*	*	3.5M	18 829.4
webbase-2001	118M	51.7M	13.0	17.3M	121.1	0.7M	4 207.8
asia.osm	12M	626.7K	0.8	594.4K	1.4	15.2K	204.7
road_usa	24M	2.5M	2.5	2.4M	4.1	0.2M	310.0
europe.osm	51M	1 500.0K	4.1	1 329.9K	6.1	8.4K	302.4
rgg26	67M	67.1M	1.0	51.3M	172.6	49.6M	9 887.7
rhg	100M	*	*	*	*	0	124.0
del24	17M	16.8M	0.2	15.6M	12.7	12.4M	4 789.5
del26	67M	67.1M	0.7	62.5M	53.3	49.9M	20 728.7

Scalable Reductions [Hespe et al. 2018]

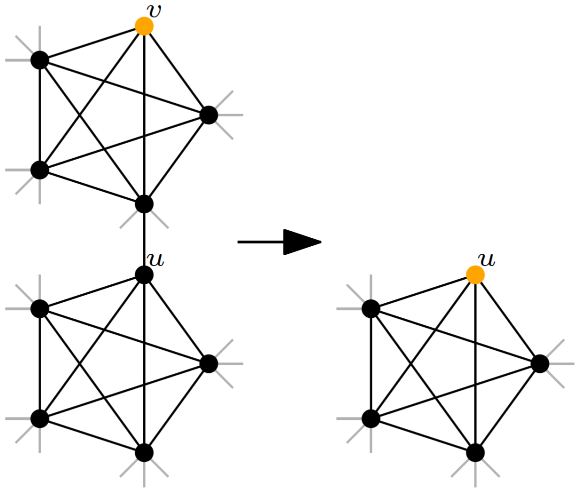
Problem: Effective reductions are slow

Graph		LinearTime		NearLinear		VCSolver	
name	n	$\|\mathcal{K}\|$	time	$\|\mathcal{K}\|$	time	$\|\mathcal{K}\|$	time
uk-2002	19M	11.7M	1.5	4.0M	28.0	0.2M	336.9
arabic-2005	23M	15.6M	2.6	6.7M	246.1	0.6M	1 033.2
gsh-2015-tpd	31M	2.0M	11.6	1.2M	97.4	0.4M	372.3
uk-2005	39M	28.2M	2.5	5.9M	60.5	0.8M	541.4
it-2004	41M	27.1M	3.3	11.3M	1 544.6	1.6M	6 749.0
sk-2005	51M	*	*	*	*	3.2M	10 010.5
uk-2007-05	106M	*	*	*	*	3.5M	18 829.4
webbase-2001	118M	51.7M	13.0	17.3M	121.1	0.7M	4 207.8
asia.osm	12M	626.7K	0.8	594.4K	1.4	15.2K	204.7
road_usa	24M	2.5M	2.5	2.4M	4.1	0.2M	310.0
europe.osm	51M	1 500.0K	4.1	1 329.9K	6.1	8.4K	302.4
rgg26	67M	67.1M	1.0	51.3M	172.6	49.6M	9 887.7
rhg	100M	*	*	*	*	0	124.0
del24	17M	16.8M	0.2	15.6M	12.7	12.4M	4 789.5
del26	67M	67.1M	0.7	62.5M	53.3	49.9M	20 728.7

Scalable Reductions [Hespe et al. 2018]

Solutions:

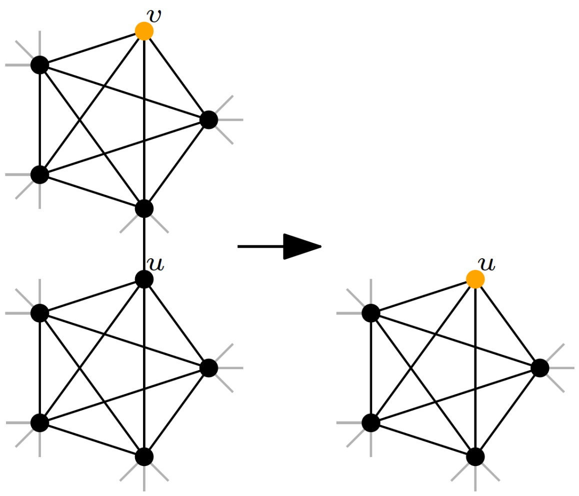
Only check parts of graph that change



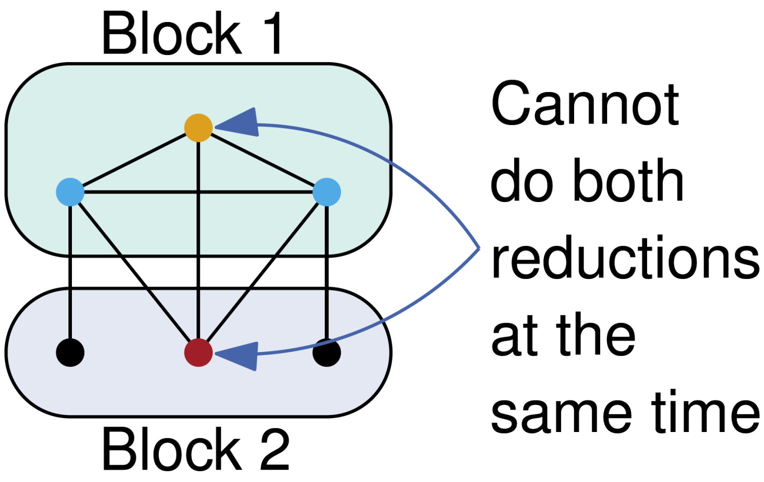
Scalable Reductions [Hespe et al. 2018]

Solutions:

Only check parts of graph that change



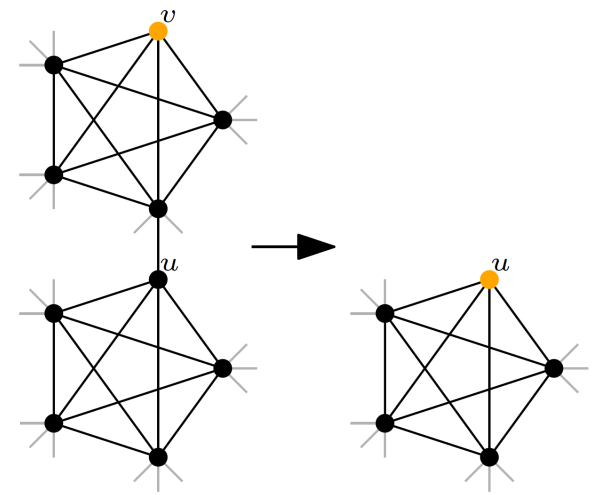
Parallelize



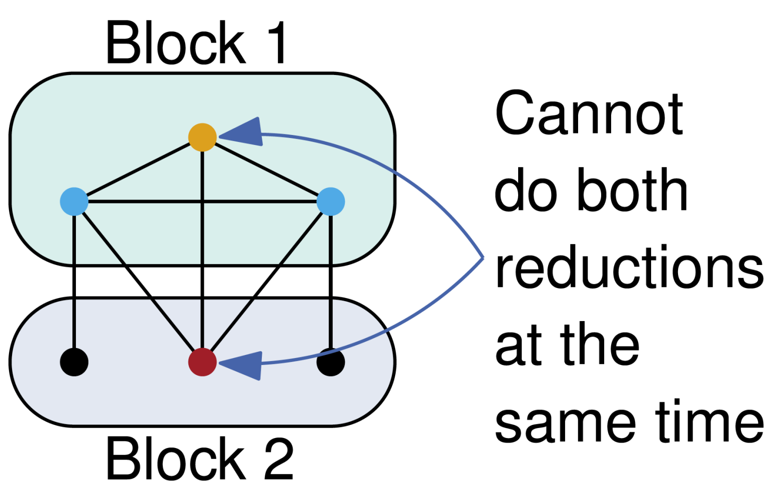
Scalable Reductions [Hespe et al. 2018]

Solutions:

Only check parts of graph that change



Parallelize



Stop reductions if they are ineffective

Scalable Reductions [Hespe et al. 2018]

NearLinear		VCSolver		ParFastKer		
$\vert\mathcal{K}\vert$	time	$\vert\mathcal{K}\vert$	time	$\vert\mathcal{K}\vert$	time	su
4.0M	28.0	0.2M	336.9	**0.3M**	11.8	28.4
6.7M	246.1	0.6M	1 033.2	**0.6M**	25.7	40.2
1.2M	97.4	0.4M	372.3	**0.5M**	32.0	11.7
5.9M	60.5	0.8M	541.4	**0.9M**	53.3	10.1
11.3M	1 544.6	1.6M	6 749.0	**1.7M**	151.8	44.4
*	*	3.2M	10 010.5	3.5M	178.3	56.1
*	*	3.5M	18 829.4	**3.7M**	372.4	50.6
17.3M	121.1	0.7M	4 207.8	**0.9M**	54.9	76.6
594.4K	1.4	15.2K	204.7	**34.9K**	1.2	169.8
2.4M	4.1	0.2M	310.0	**0.2M**	4.1	76.0
1 329.9K	6.1	8.4K	302.4	**14.2K**	4.9	61.3
51.3M	172.6	49.6M	9 887.7	**49.8M**	150.3	65.8
*	*	0	124.0	**16**	64.6	1.9
15.6M	12.7	12.4M	4 789.5	12.9M	51.5	93.1
62.5M	53.3	49.9M	20 728.7	51.7M	179.0	115.8

The weighted case

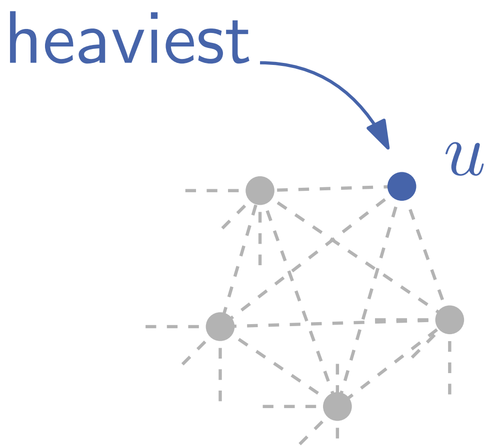
Weighted variant

> 2 months ago:

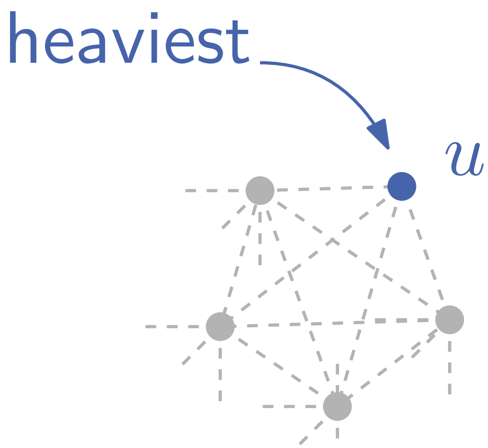
Cannot solve on graphs with 500 vertices

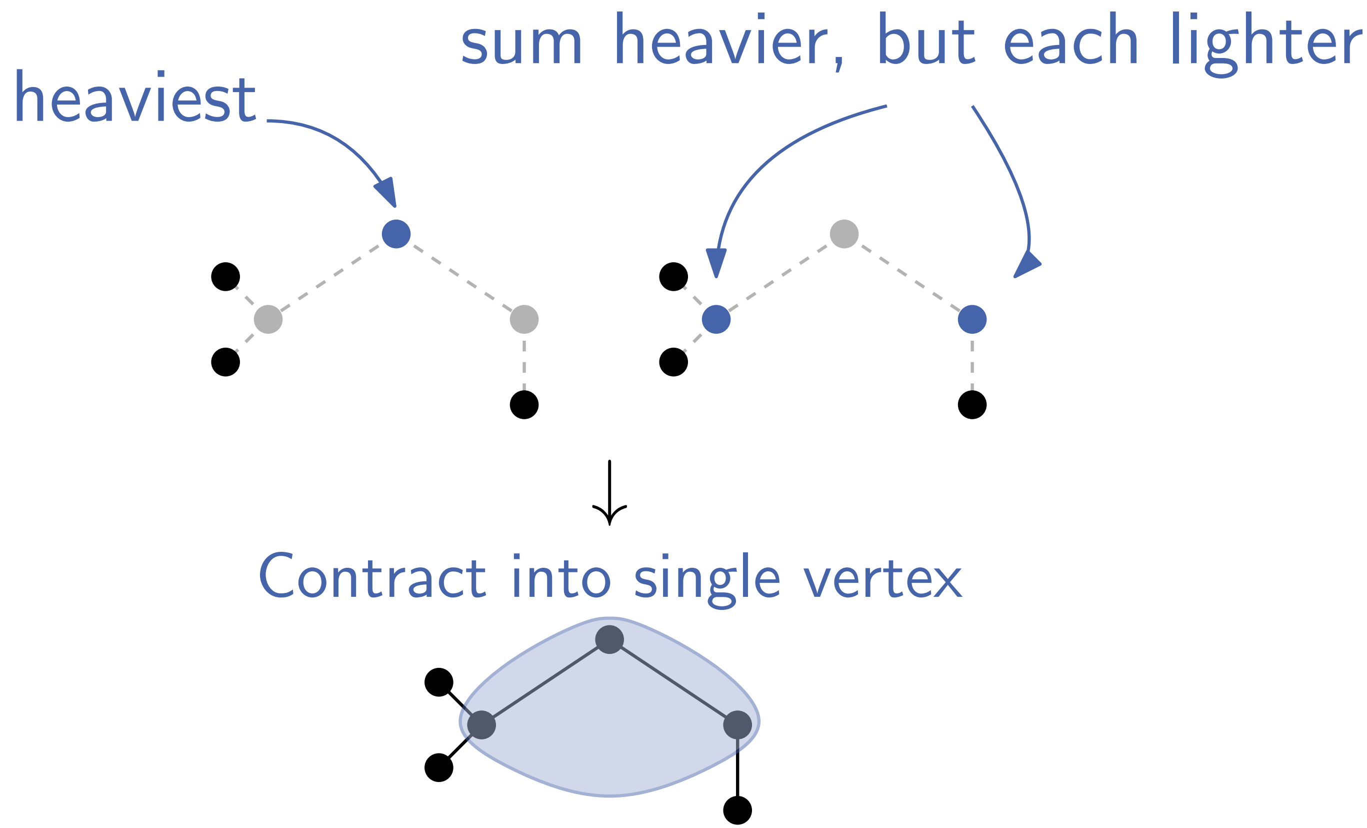
One LP reduction — untested

New reductions [Lamm et al. 2019]

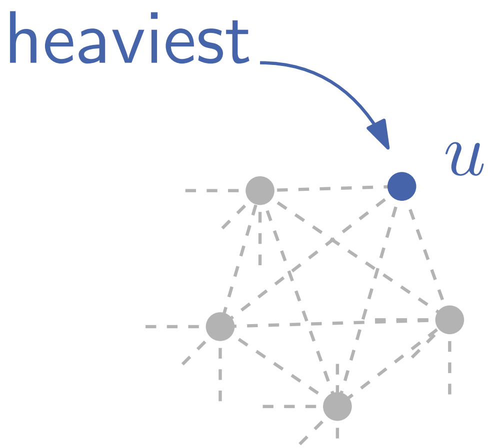


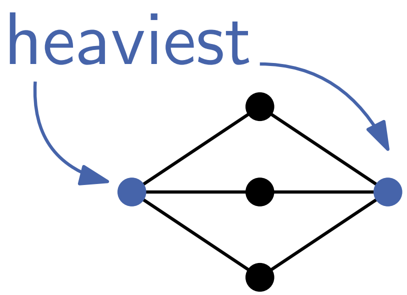
New reductions [Lamm et al. 2019]

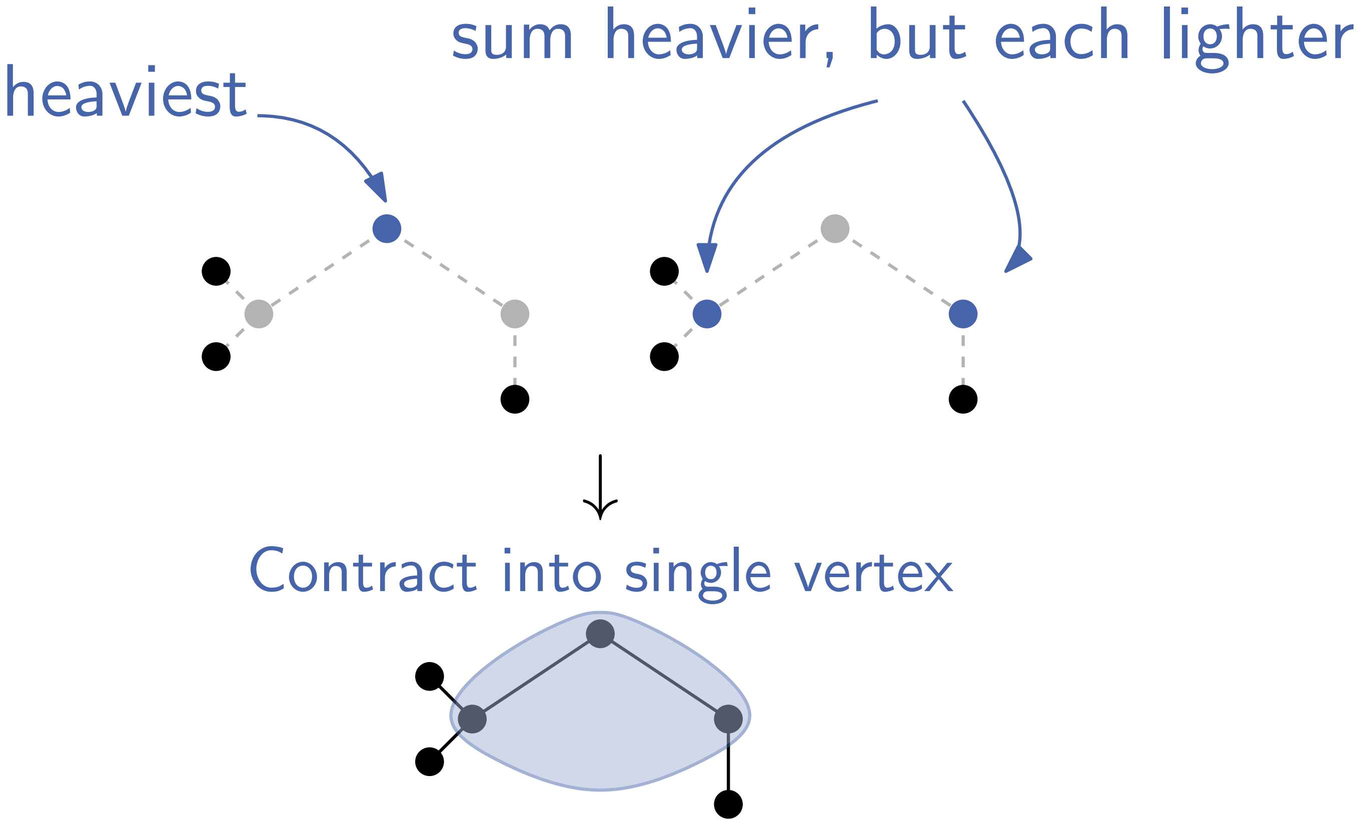




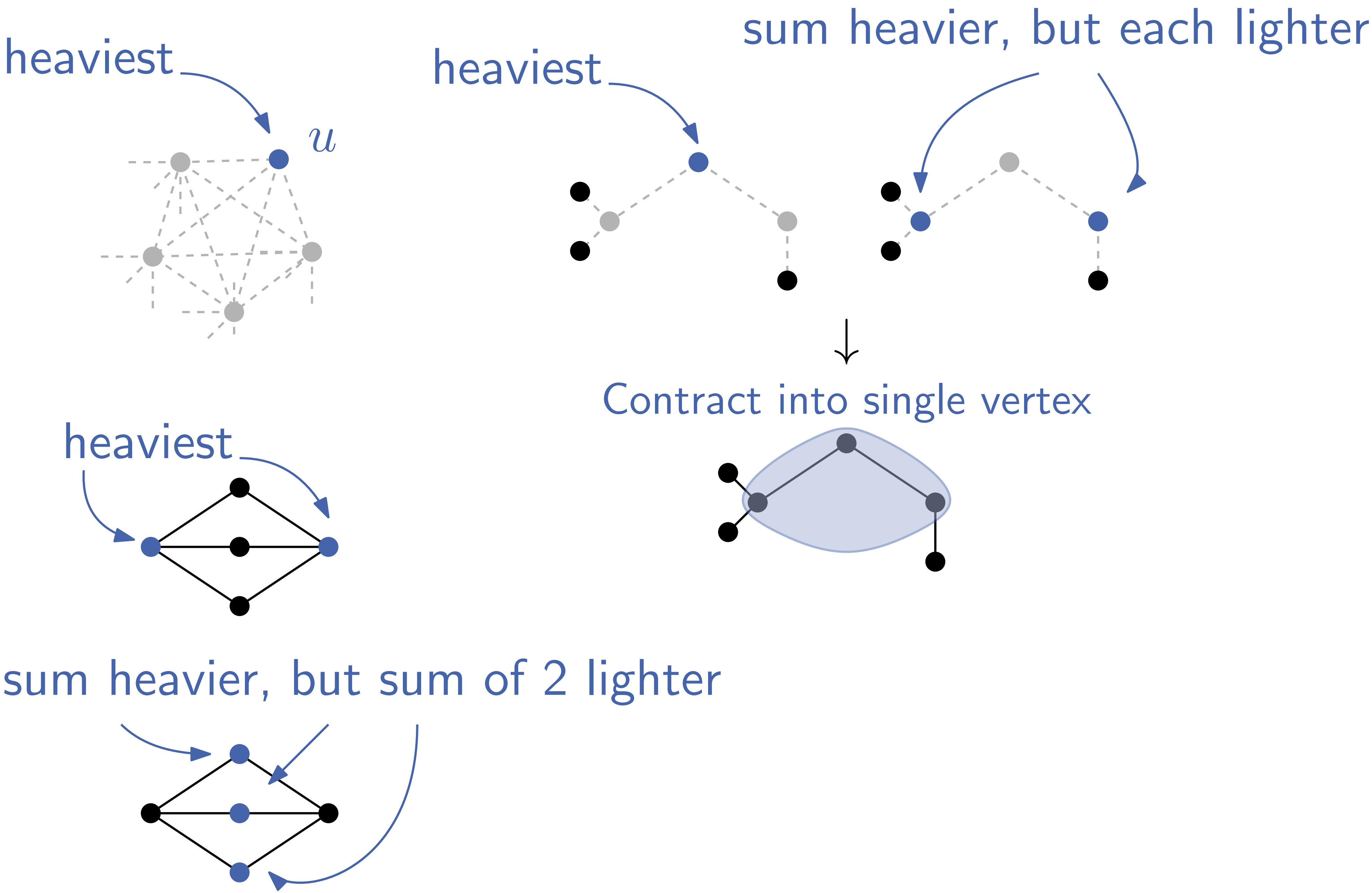
New reductions [Lamm et al. 2019]



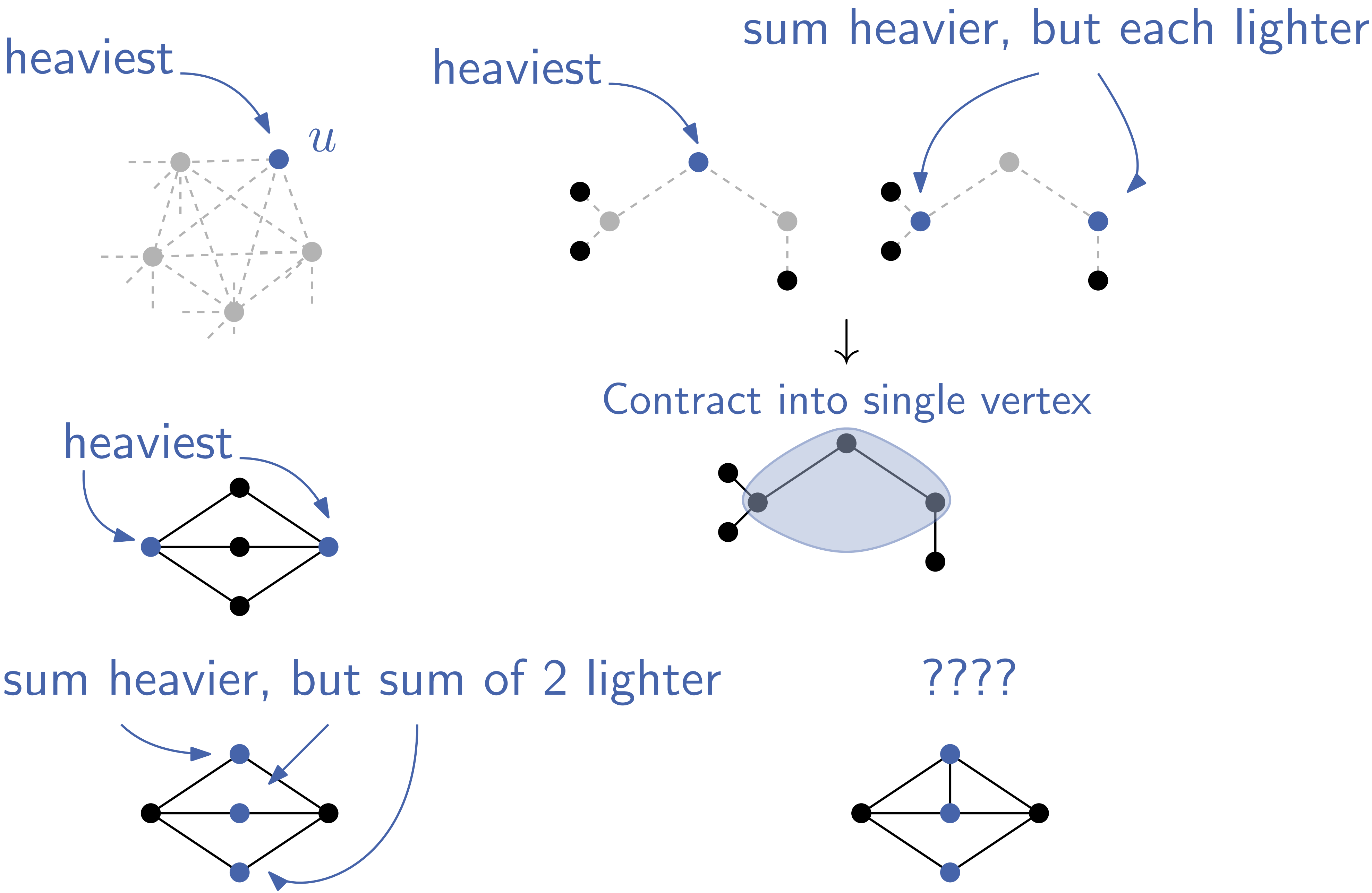




New reductions [Lamm et al. 2019]

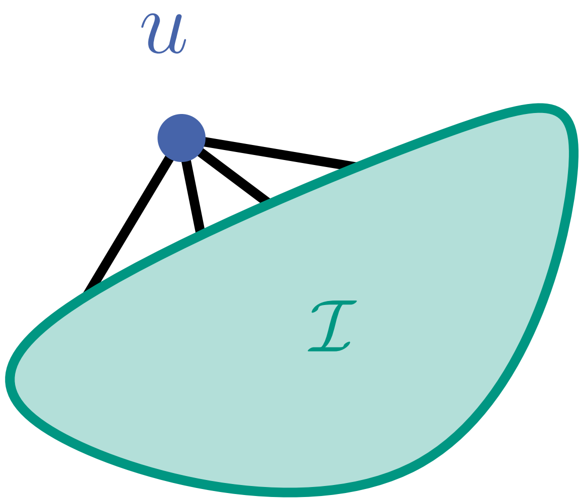


New reductions [Lamm et al. 2019]



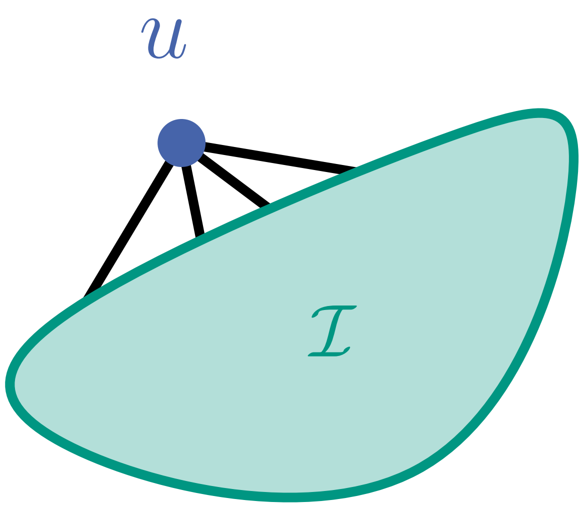
Meta-reductions

Theorem.



Meta-reductions

Theorem.

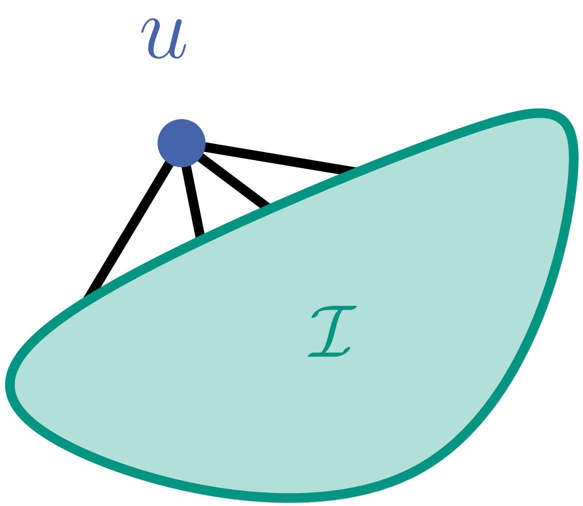


$w(u) \geq w(\mathcal{I})$?

Choose u

Meta-reductions

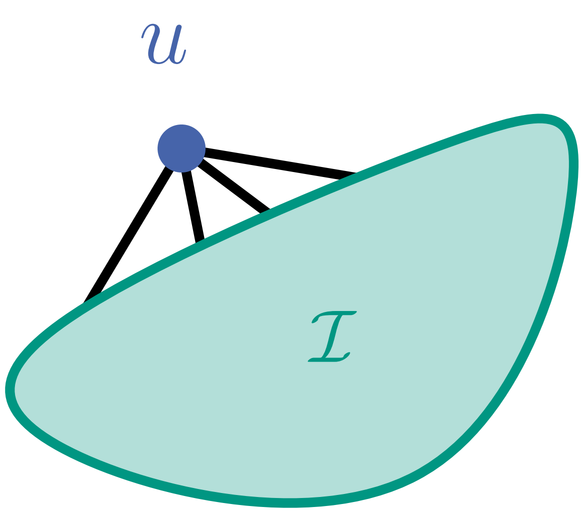
Theorem.



$w(u) \geq w(\mathcal{I})$?

Choose u

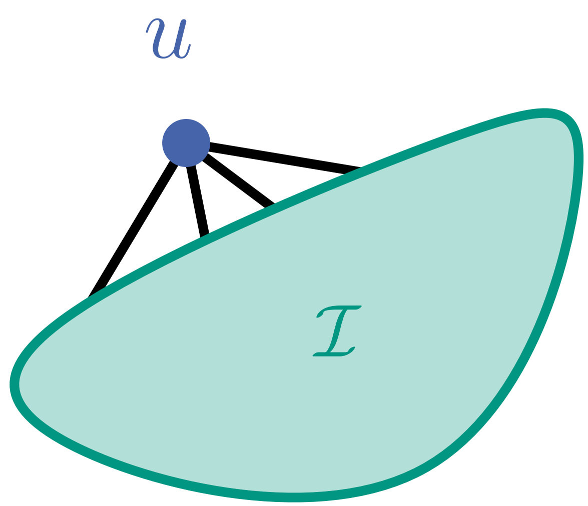
Theorem.



$w(u) < w(\mathcal{I})$?

Meta-reductions

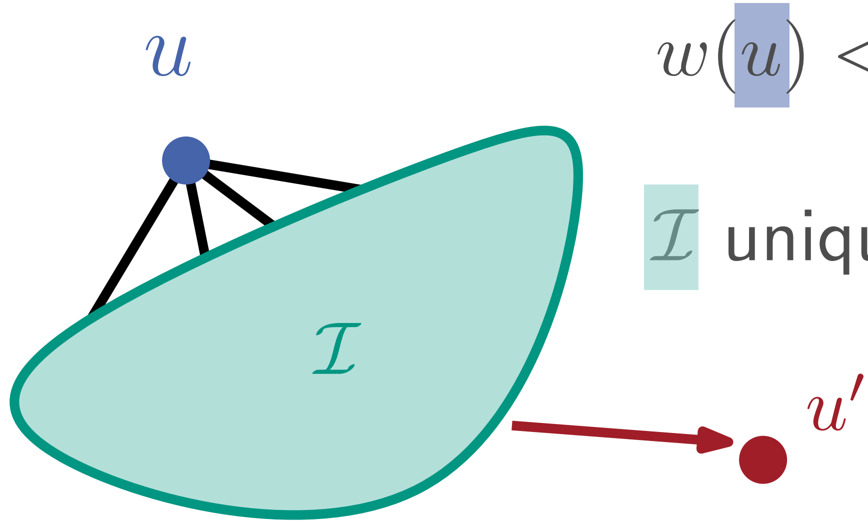
Theorem.



$w(u) \geq w(\mathcal{I})$?

Choose u

Theorem.



$w(u) < w(\mathcal{I})$?

$\mathcal{I}$ uniquely larger? contract

Practice / Application: Viability for map labeling

	B & R_{full}	
Graph	t_{max}	w_{max}
alabama-AM2	0.79	**174 309**
alabama-AM3	80.78	185 707
district-of-columbia-AM1	4.13	**196 475**
district-of-columbia-AM2	233.70	147 450
district-of-columbia-AM3	918.07	92 714
florida-AM2	0.02	**230 595**
florida-AM3	324.38	226 767
georgia-AM3	14.35	214 918
greenland-AM3	47.25	13 069
hawaii-AM2	10.89	**125 284**
hawaii-AM3	1 177.95	129 812
idaho-AM3	61.26	76 831
kansas-AM3	18.99	87 925
kentucky-AM2	42.05	**97 397**
kentucky-AM3	3 346.94	96 634
louisiana-AM3	20.17	**60 024**
maryland-AM3	11.08	**45 496**
massachusetts-AM2	0.48	**140 095**
massachusetts-AM3	23.97	145 631
mexico-AM3	289.14	**97 663**
new-hampshire-AM3	8.75	**116 060**
north-carolina-AM3	11.55	49 562
oregon-AM2	0.09	**165 047**
oregon-AM3	474.15	164 941
pennsylvania-AM3	38.76	**143 870**
rhode-island-AM2	16.79	184 543
rhode-island-AM3	931.05	163 080
utah-AM3	285.22	**98 847**
vermont-AM3	443.88	55 577
virginia-AM2	0.77	**295 867**
virginia-AM3	786.05	233 572
washington-AM2	2.20	**305 619**
washington-AM3	532.25	271 747
west-virginia-AM3	854.73	**47 927**

Practice / Application: Viability for map labeling

	B & R$_{\text{full}}$	
Graph	$t_{\max}$	$w_{\max}$
alabama-AM2	0.79	**174 309**
alabama-AM3	80.78	185 707
district-of-columbia-AM1	4.13	**196 475**
district-of-columbia-AM2	233.70	147 450
district-of-columbia-AM3	918.07	92 714
florida-AM2	0.02	**230 595**
florida-AM3	324.38	226 767
georgia-AM3	14.35	214 918
greenland-AM3	47.25	13 069
hawaii-AM2	10.89	**125 284**
hawaii-AM3	1 177.95	129 812
idaho-AM3	61.26	76 831
kansas-AM3	18.99	87 925
kentucky-AM2	42.05	**97 397**
kentucky-AM3	3 346.94	96 634
louisiana-AM3	20.17	**60 024**
maryland-AM3	11.08	**45 496**
massachusetts-AM2	0.48	**140 095**
massachusetts-AM3	23.97	145 631
mexico-AM3	289.14	**97 663**
new-hampshire-AM3	8.75	**116 060**
north-carolina-AM3	11.55	49 562
oregon-AM2	0.09	**165 047**
oregon-AM3	474.15	164 941
pennsylvania-AM3	38.76	**143 870**
rhode-island-AM2	16.79	184 543
rhode-island-AM3	931.05	163 080
utah-AM3	285.22	**98 847**
vermont-AM3	443.88	55 577
virginia-AM2	0.77	**295 867**
virginia-AM3	786.05	233 572
washington-AM2	2.20	**305 619**
washington-AM3	532.25	271 747
west-virginia-AM3	854.73	**47 927**

and more graphs

	B & R$_{\text{full}}$	
Graph	$t_{\max}$	$w_{\max}$
as-skitter	746.93	**123 904 741**
ca-AstroPh	0.03	**796 556**
ca-CondMat	0.02	**1 143 480**
ca-GrQc	0.00	**289 481**
ca-HepPh	0.02	**579 675**
ca-HepTh	0.01	**560 662**
email-Enron	0.03	**2 457 547**
email-EuAll	0.19	**25 330 331**
p2p-Gnutella04	0.01	**667 539**
p2p-Gnutella05	0.01	**556 559**
p2p-Gnutella06	0.01	**547 591**
p2p-Gnutella08	0.01	**435 893**
p2p-Gnutella09	0.01	**568 472**
p2p-Gnutella24	0.02	**1 970 329**
p2p-Gnutella25	0.02	**1 697 310**
p2p-Gnutella30	0.03	**2 785 957**
p2p-Gnutella31	0.04	**4 750 671**
roadNet-CA	774.56	**111 408 830**
roadNet-PA	32.06	**61 686 106**
roadNet-TX	33.49	**78 606 965**
soc-Epinions1	0.11	**5 668 401**
soc-LiveJournal1	270.96	**283 948 671**
soc-Slashdot0811	0.18	**5 650 791**
soc-Slashdot0902	0.21	**5 953 582**
soc-pokec-relationships	1 404.57	75 717 984
web-BerkStan	831.75	**43 766 431**
web-Google	3.16	**56 313 384**
web-NotreDame	28.11	25 957 800
web-Stanford	4.69	**17 799 469**
wiki-Talk	3.36	**235 875 181**
wiki-Vote	0.06	**500 436**

Conclusion

Reduction efficiency is important in practice

Reductions are effective in practice

Reductions + heuristics are a winning combination

Next? $\rightarrow$ transfer to theory

Conclusion

Reduction efficiency is important in practice

Reductions are effective in practice

Reductions + heuristics are a winning combination

Next? $\rightarrow$ transfer to theory

Thank you!