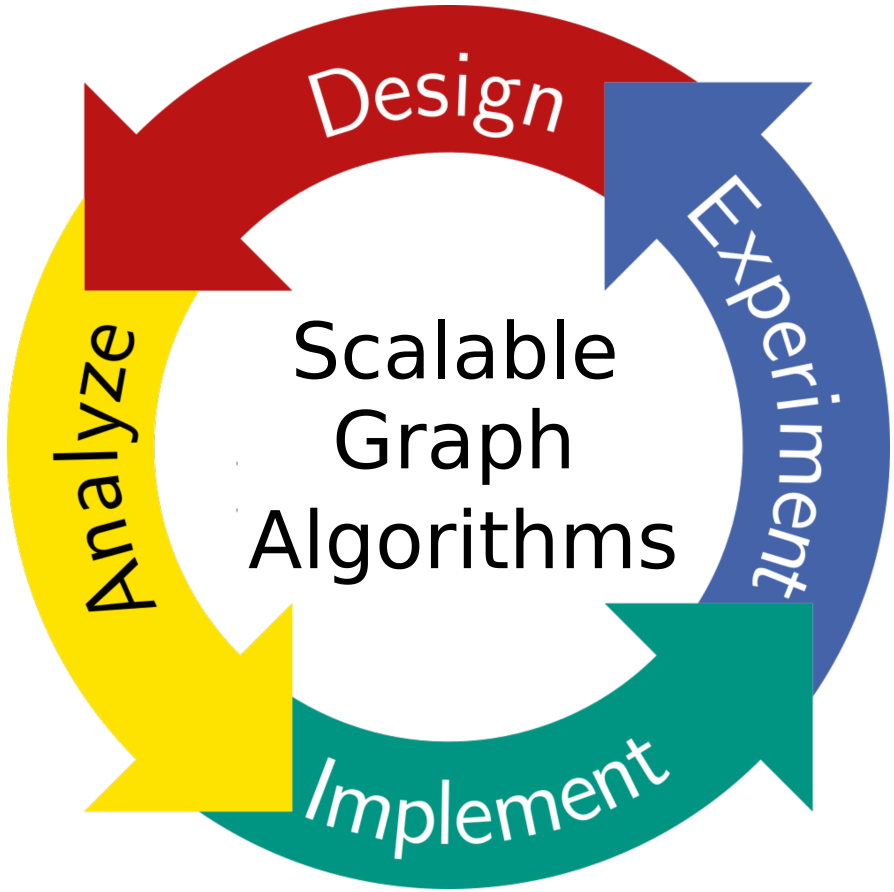

Design
Experiment
Implement
Analyze
Scalable
Graph
Algorithms

Curriculum Vitae

Education/Awards

Studies at KIT/Univie 2004–2013:

- Diploma mathematics (1.0)
- Diploma informatics (1.1)
- PhD summa cum laude

Honors and Awards:

- Best Student of Department of Informatics (KIT)
- Uniserv Dissertation Award
- KIT Doctoral Award in the Area of Competence "Information, Communication and Organization"
- Heinz Billing Prize
- Best Paper Award IPDPS'18
- SPP Algorithms for Big Data: Best Paper Award
- PACE Implementation Challenge

Curriculum Vitae

Professional Experience

January 2017 – present	UNIVERSITY VIENNA. Research associate in group of Monika Henzinger *Leading Algorithm Engineering Subgroup*
March 2010 – December 2017 since 2013:	KARLSRUHE INSTITUTE OF TECHNOLOGY. Research associate in group of Peter Sanders *Leading Parallel Algorithms & Partitioning Subgroup*
2015/2016	TU VIENNA visiting scientist/professor in the group of Jesper Larsson Träff

Curriculum Vitae

Third Party Funding / Professional Service

January 2017 — PARTITIONING LARGE GRAPHS
Co-PIs: Dorothea Wagner, Peter Sanders
DFG application. Funded: 580T Euro.

Sept 2017 — PROCESS MAPPING
Co-PI: Jesper Larsson Träff
FWF application. Funded: 312T Euro.

Sept 2019 — KERNELIZATION
FWF Start application submitted

PC Member:
ESA'17, IPDPS'18, IPDPS'19, IPEC'19, ISC'19, ALENEX'20, CSC'20 [...]

Reviewer:
JEA, ALX, SPAA, IPDPS, DIMACS, ESA, TPDS, JPDC, SODA, TVCG, TOPC, SISC [...]

Curriculum Vitae

Third Party Funding / Professional Service

January 2017 — PARTITIONING LARGE GRAPHS
Co-PIs: Dorothea Wagner, Peter Sanders
DFG application. Funded: 580T Euro.

Sept 2017 — PROCESS MAPPING
Co-PI: Jesper Larsson Träff
FWF application. Funded: 312T Euro.

Sept 2019 — KERNELIZATION
FWF Start application submitted

Co-organizer Shonan Meeting
"Parameterized algorithms & data reduction: Theory meets practice"

Guest Editor Special Issue of Algorithms
"Graph Partitioning: Theory, Engineering and Applications".

Invited Speaker Finse Winter School on "Parameterized Complexity"

Curriculum Vitae

Teaching

Supervision:

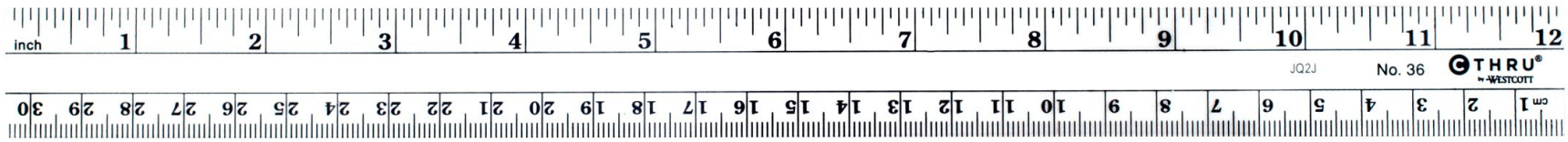
2010-19	$\geq$30 thesis (BA/MA), 3 currently running

Lecturing:

2014-17	5xGraph Partitioning and Clustering, KIT
2016	Algorithms II, KIT
2017-18	Advanced Algorithms, Uni Vie
2018	Algorithms and Data Structures, Uni Vie
2018	Algorithms and Data Structures II, Uni Vie
2019	Numerical Algorithms, Uni Vie
2019	Algorithms and Data Structures II, Uni Vie
2019	Distributed and Parallel Algorithms, Uni Vie
2019/20	Numerical Algorithms, Uni Vie
2019/20	Advanced Topics in Algorithms, Uni Vie

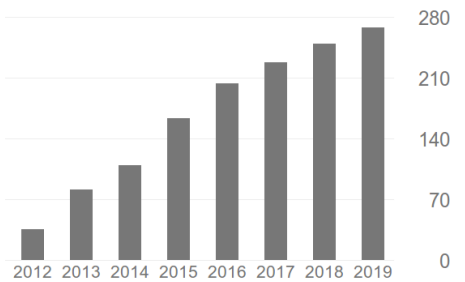
$\sum \rightarrow$ 30 SWS

Metrics

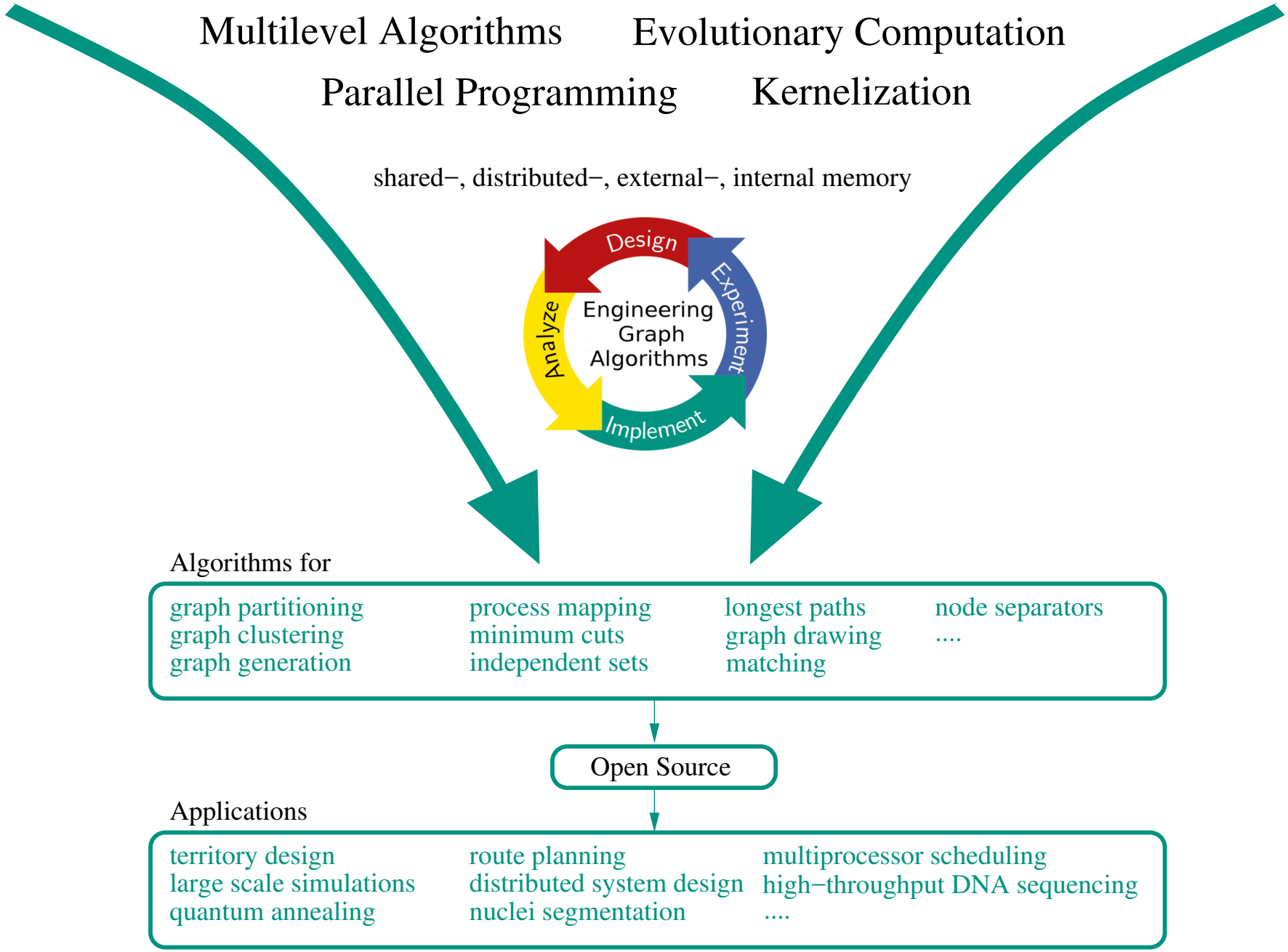


- six years after PhD
- open source: Ka{HIP, Gen, MIS, Draw, LP}, VieM, VieClus, VieCut
- 58 co-authors
- 57 peer-reviewed publications (+4 currently in submission)
  - Algorithm Engineering
    (12x ALX, 1x CSC, 1x ESA, 12x SEA, 3x GECCO, 1x SoCS)
  - Parallel Processing (4x IPDPS, 3x Euro-Par, 1x SPAA)
  - 42 publications as PostDoc
  - 4 book chapters
  - 10 journal papers (JEA, JoH, TPDS, TVCG, JPDC)

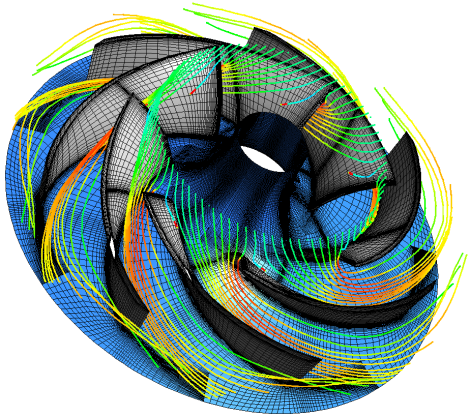
	All	Since 2014
Citations	1394	1238
h-index	17	16
i10-index	22	20



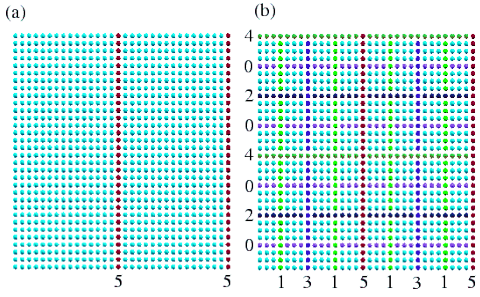
Research Overview

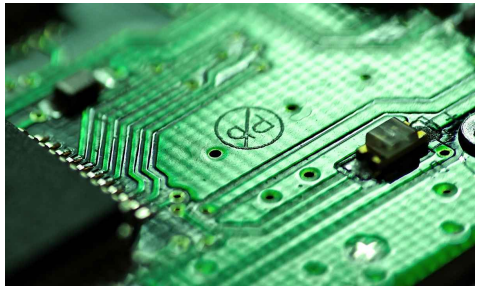

Multilevel Algorithms
Evolutionary Computation
Parallel Programming
Kernelization
shared−, distributed−, external−, internal memory
Design
Experiment
Implement
Analyze
Engineering Graph Algorithms
Algorithms for
graph partitioning
graph clustering
graph generation
process mapping
minimum cuts
independent sets
longest paths
graph drawing
matching
node separators
....
Open Source
Applications
territory design
large scale simulations
quantum annealing
route planning
distributed system design
nuclei segmentation
multiprocessor scheduling
high−throughput DNA sequencing
....

Applications



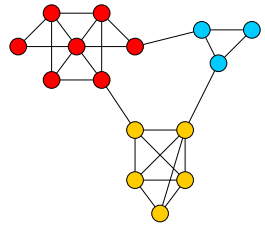
$$\mathbf{R}^{n \times n} \ni Ax = b \in \mathbf{R}^{n}$$





Highlights

Graph Clustering

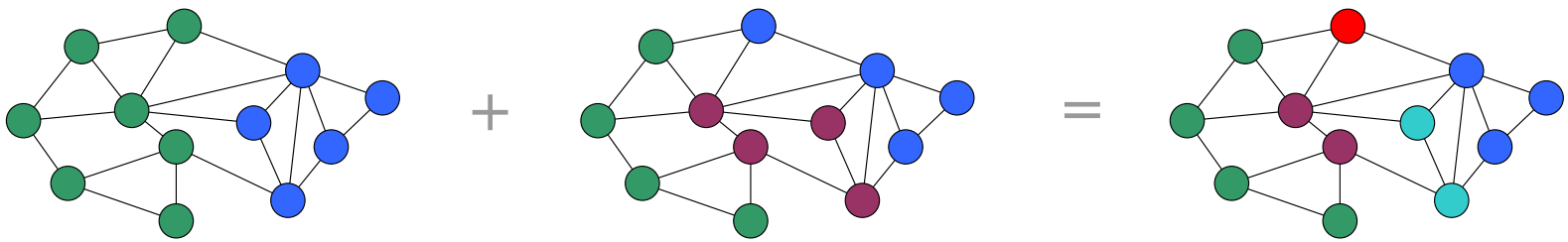


Problem:

- partition graph into tightly connected groups
- clustering paradigm: internally dense and externally sparse
- quality measure: modularity

Recombination Mechanism

- borrowed from ML: ensemble learning → maximum overlap

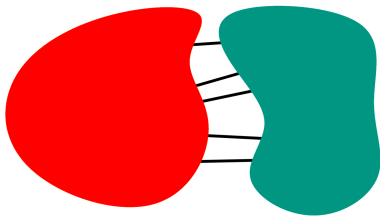


→ results in overlap clustering, contract + recluster
→ + coarse-grained parallelization

Outperform all DIMACS Challenge results
with ONE solver and in less time

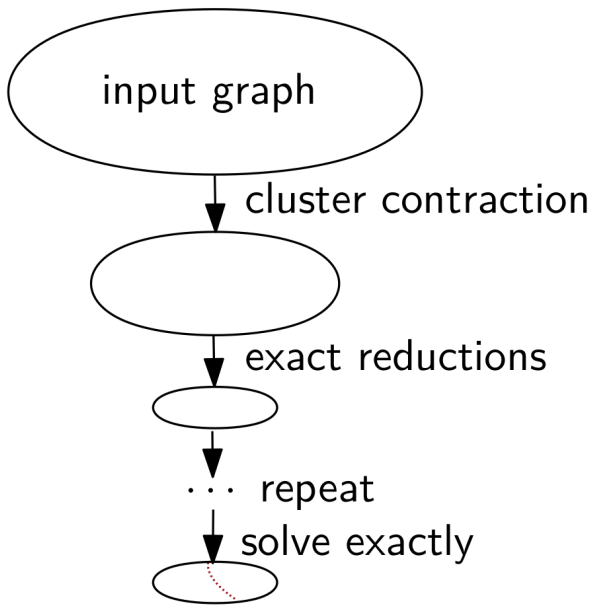
Minimum Cuts

Cut: A cut in a multigraph is a partition of $V = C \cup \overline{C}$
Problem: size of minimum cut in G?



Contributions:

- (in)exact reductions + solve kernel to optimality
  $\rightarrow$ linear running time, but potentially suboptimal cuts
- NO guarantee, but experiments say likely opt
- reductions depend on bound $\hat{\lambda}$
  $\rightsquigarrow$ use $\hat{\lambda}$ in exact (parallel) NOI algorithm
  $\rightsquigarrow$ fastest exact minimum cut algorithm
  $\approx$ order of magnitude



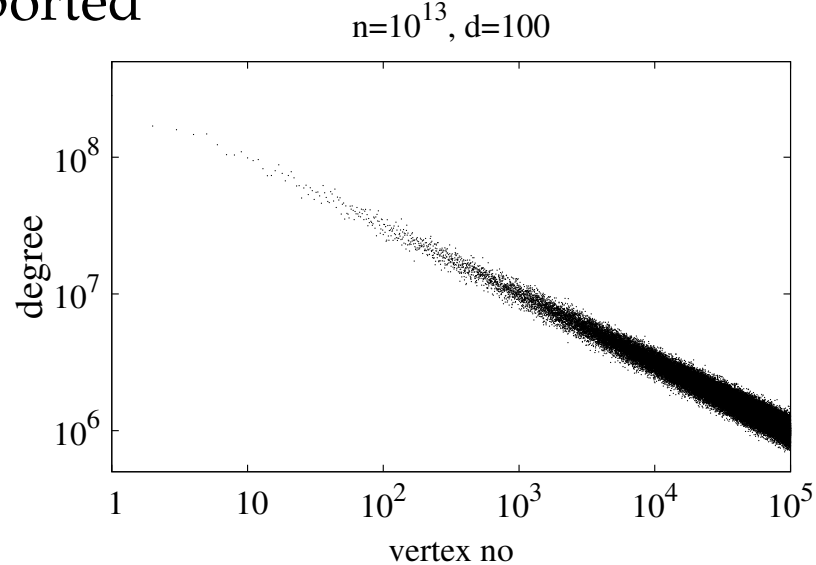
Graph Generation

Ideas:

- replace randomness by pseudo-randomness, e.g. hashing
- replace array accesses by recomputation
  $\rightarrow$ embarrassingly parallel algorithm without communication

Results:

- generate Peta-edge BA graph in $<$ one hour (10^{15} edges, 16K cores)
- 20 000 times larger than previously reported
- 16x faster than RMAT ($m = 50 \cdot 10^9$)
- communication efficient algorithms



more work:

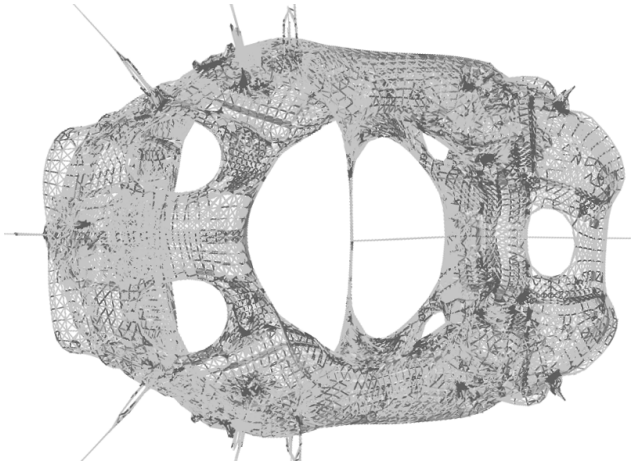
- Barabasi-Albert
- more models:
  ER, G(n,m), G(n,p),
  RHG, RGG, DEL

Graph Drawing

$$G = (V, E, d)$$
$$d : E \to \mathbf{R}$$

$\Rightarrow$

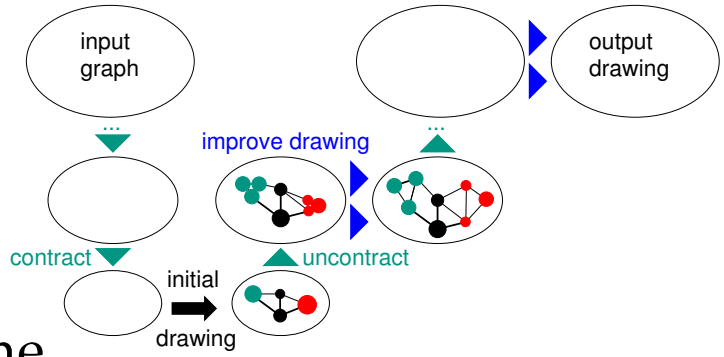
$$x : V \to \mathbf{R}^2$$



Maximal Entropy Stress Model:

$$\max H(x) := \sum_{\{u,v\} \notin E} \ln ||x_u - x_v||$$

$$\text{subject to } ||x_u - x_v|| = d_{uv}, \{u, v\} \in E$$

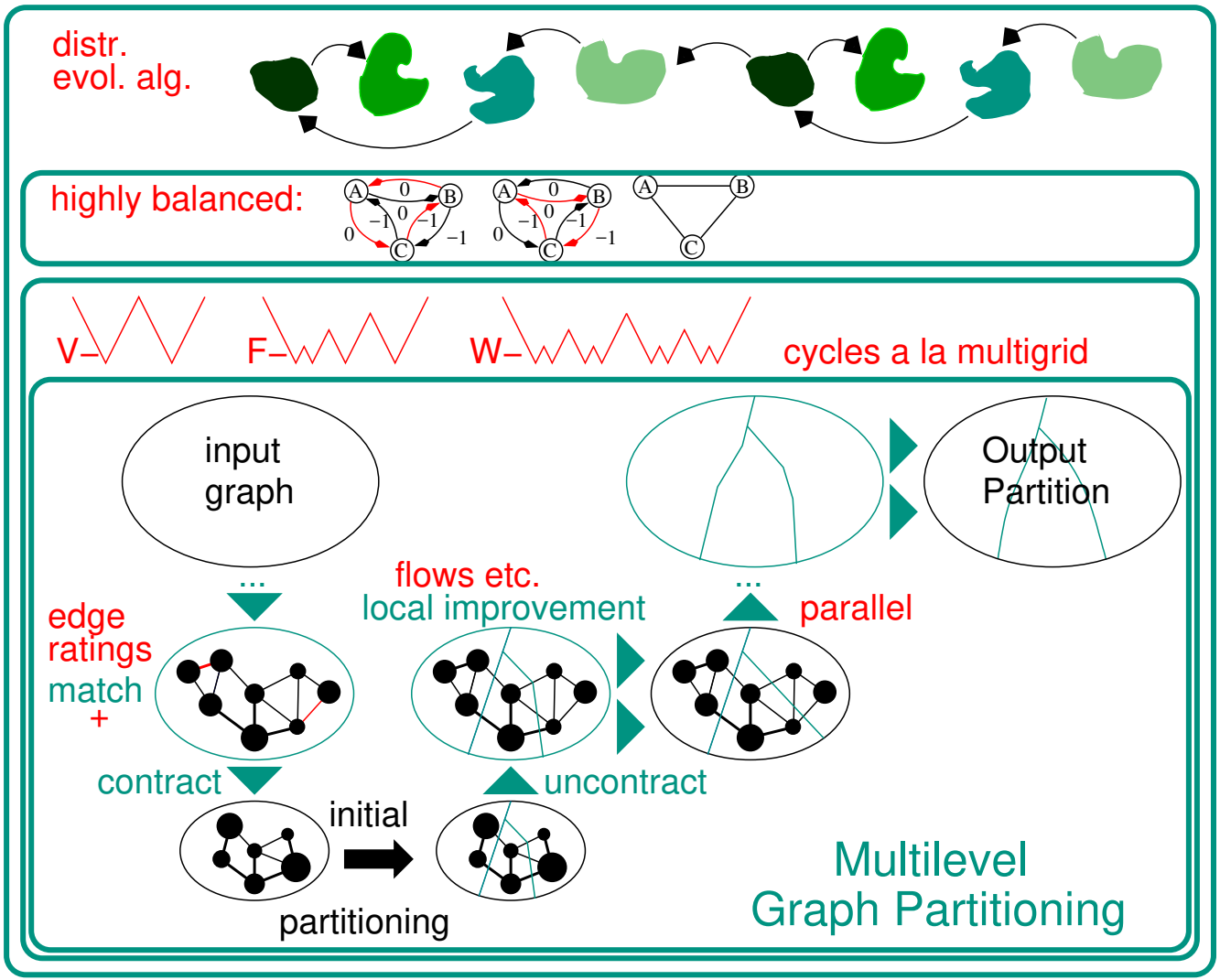


Contributions:

- multilevel integration of iterative scheme
- approximate long-range forces, by cluster contraction
- employ parallelism

Draw graphs with millions of vertices in seconds.

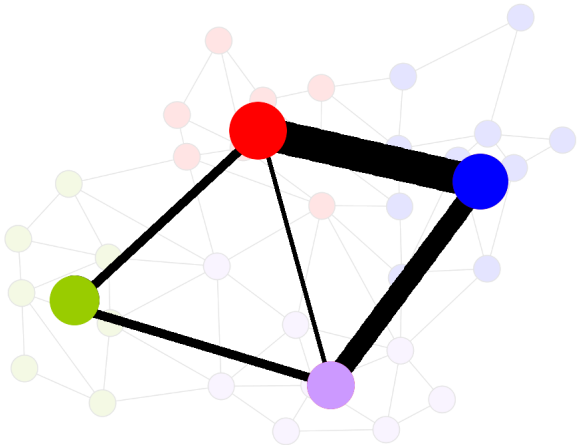
Karlsruhe High Quality Partitioning



Karlsruhe High Quality Partitioning

- pre 2014:
  - multi-level, localized and flow-based local search
  - perfectly balanced,
  - distributed evolutionary algorithms
  - specialized algorithms for road networks
- social networks
- distributed memory
- semi-external memory
- hypergraph partitioning
- node separators
- DAG partitioning
- shared-memory parallel
- ILP-based

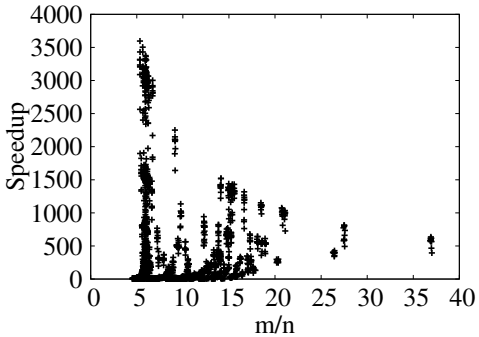
Process Mapping



$\rightarrow ?$

Exploiting Assumptions:

- communication matrix $\mathcal{C}$ is sparse $\rightarrow$ graph $G_{\mathcal{C}}$
- compute system is hierarchically structured



Yields:

- three orders of magnitude faster local search
- 50% improved mapping quality over state-of-the-art

$\rightarrow$ resulted in a FWF project

Support Vector Machines

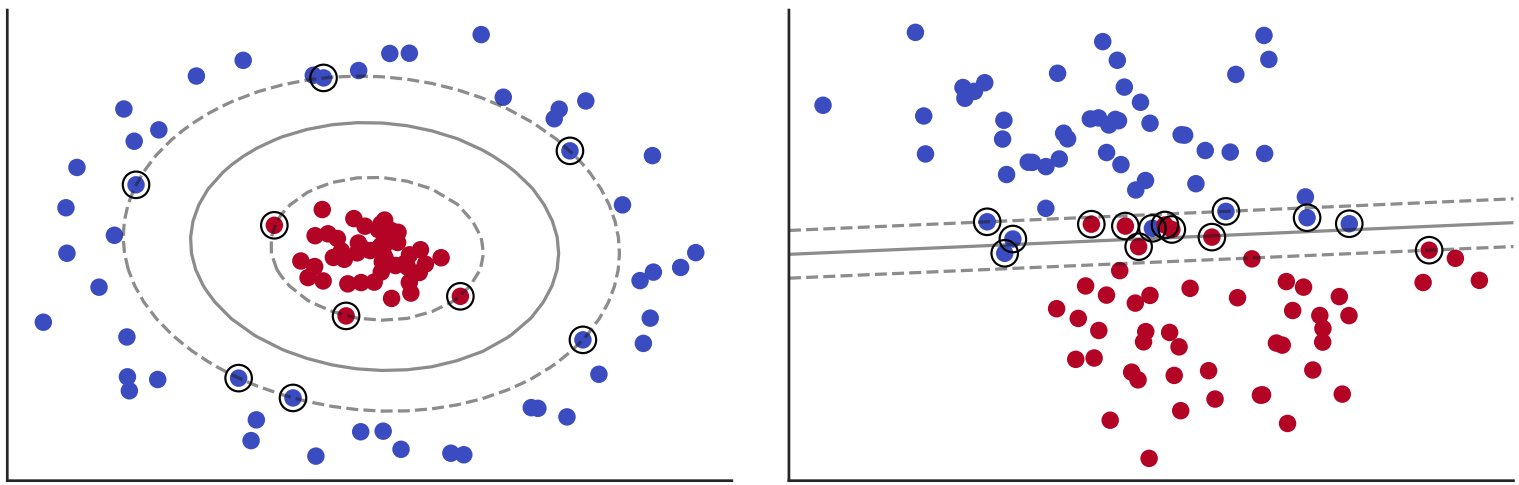
- **Binary Classification Problem**:

  Train classifier on n labeled data points $(x_i \in \mathbf{R}^d, y_i \in \{-1, 1\})$
  Goal: Assign label y_{n+1} to *new* data points x_{n+1}

- **SVM Optimization Problem:**

$$\text{min.} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad y_i(w \cdot \phi(x_i) - b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$



Support Vector Machines

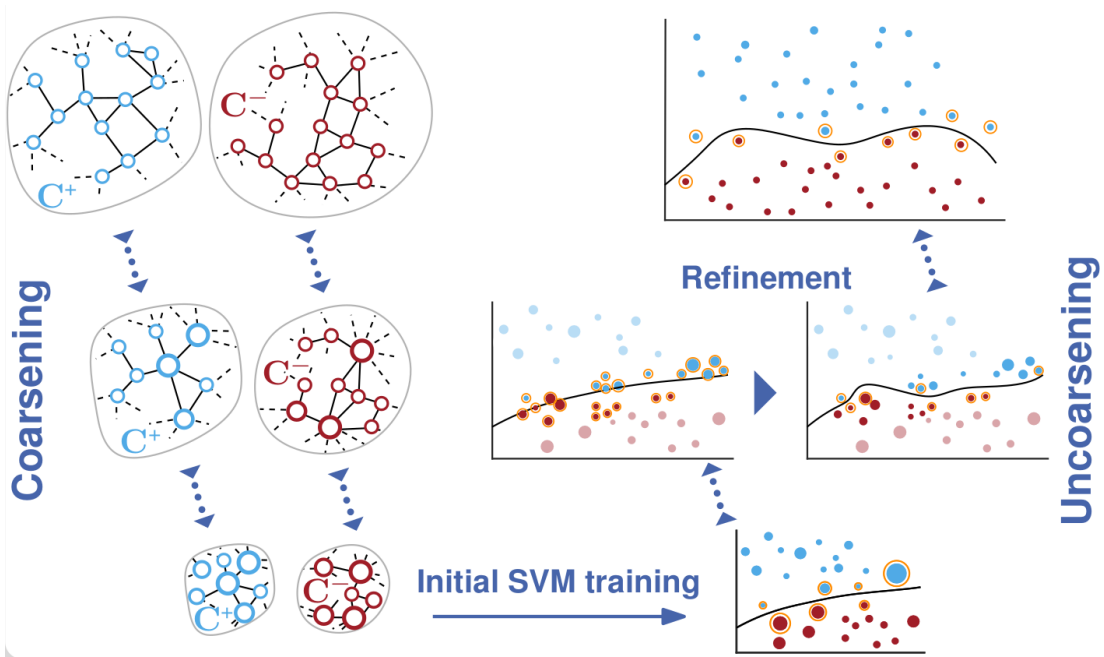
- **Binary Classification Problem**:

  Train classifier on n labeled data points $(x_i \in \mathbf{R}^d, y_i \in \{-1, 1\})$
  Goal: Assign label y_{n+1} to *new* data points x_{n+1}

- **SVM Optimization Problem:**

$$\text{min.} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad y_i(w \cdot \phi(x_i) - b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$



Support Vector Machines

- **Binary Classification Problem**:

  Train classifier on n labeled data points $(x_i \in \mathbf{R}^d, y_i \in \{-1, 1\})$
  Goal: Assign label y_{n+1} to *new* data points x_{n+1}

- **SVM Optimization Problem:**

$$\text{min.} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.} \quad y_i(w \cdot \phi(x_i) - b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

multilevel SVM using label propagation
$\rightarrow$ comparable classification quality
$\rightarrow$ up to **two** orders of magnitude faster training

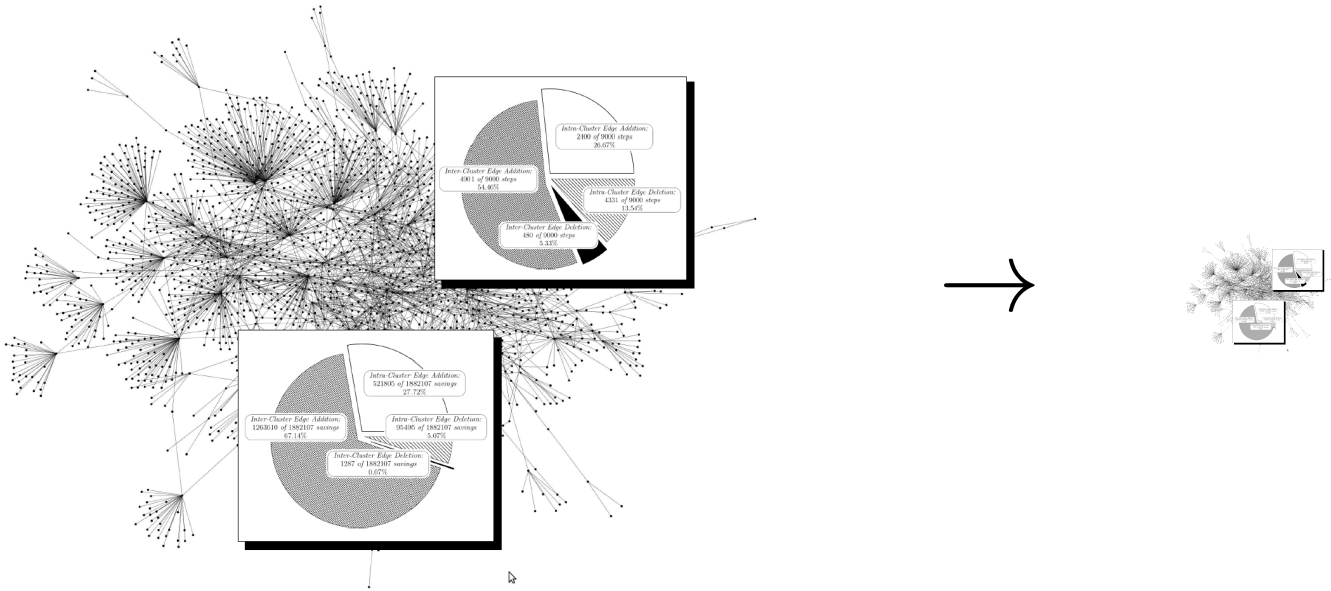
currently working on parallelization

Concrete Example
Applied Kernelization for Independent Sets

Kernelization

General Idea

Reductions:
rules to decrease graph size, while maintaining optimality

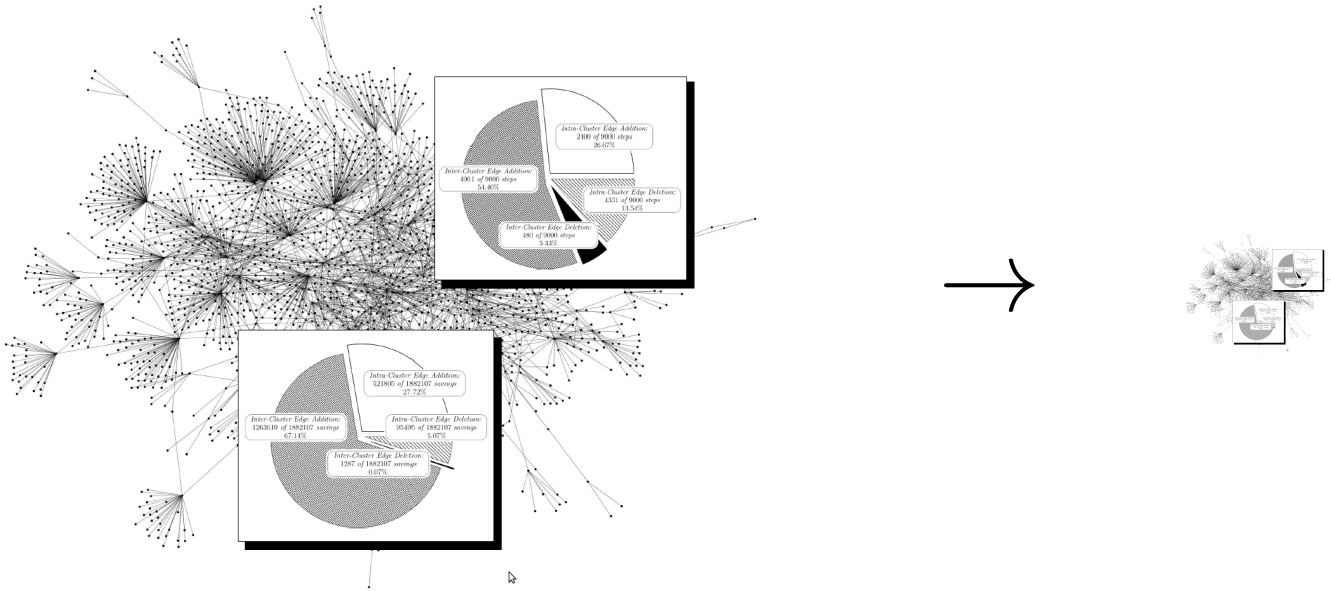


solve problem on problem kernel
$\rightarrow$ obtain solution on input graph

Kernelization

General Idea

Reductions:
rules to decrease graph size, while maintaining optimality

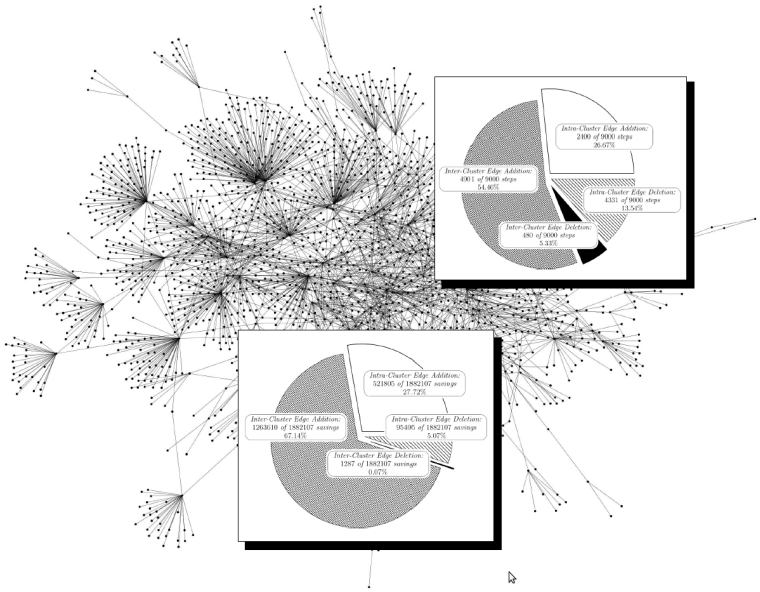


solve problem on problem kernel (using a heuristic)
$\rightarrow$ obtain solution on input graph quickly

Kernelization

General Idea

Reductions:
rules to decrease graph size, while maintaining optimality



solve problem on problem kernel
→ obtain solution on input graph

Independent Sets

- evolutionary [SEA'15]
- reduction + evolutionary [ALX'16]
- online reductions + LS [SEA'16]
- shared-mem parallel [ALX'18]
- weighted exact [ALX'19]

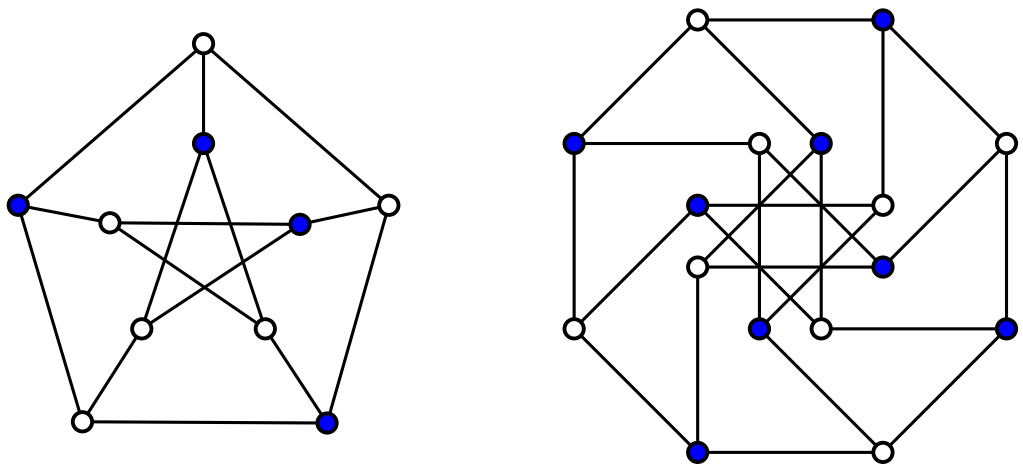
Definitions

Independent Set:

- subset $S \subseteq V$ such that there are no adjacent nodes in S

Maximum Independent Set (MIS):

- maximum cardinality set S

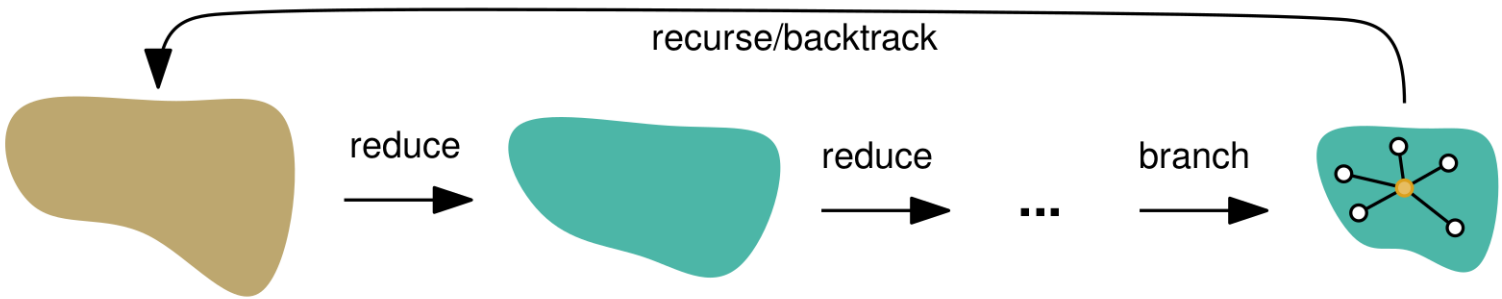


- related to maximum clique and minimum vertex cover
- finding a MIS is **NP-hard** and hard to approximate

Exact Algorithms
Independent Sets

Exact solution using **branch and bound**:

- Modify and remove subgraphs during recursions $\rightarrow$ **reductions**
- Branch when the graph can no longer be reduced



$\rightarrow$ Running time $O(1.211^n)$ [Akiba, Iwata'16]

Works well, except when it doesn't:

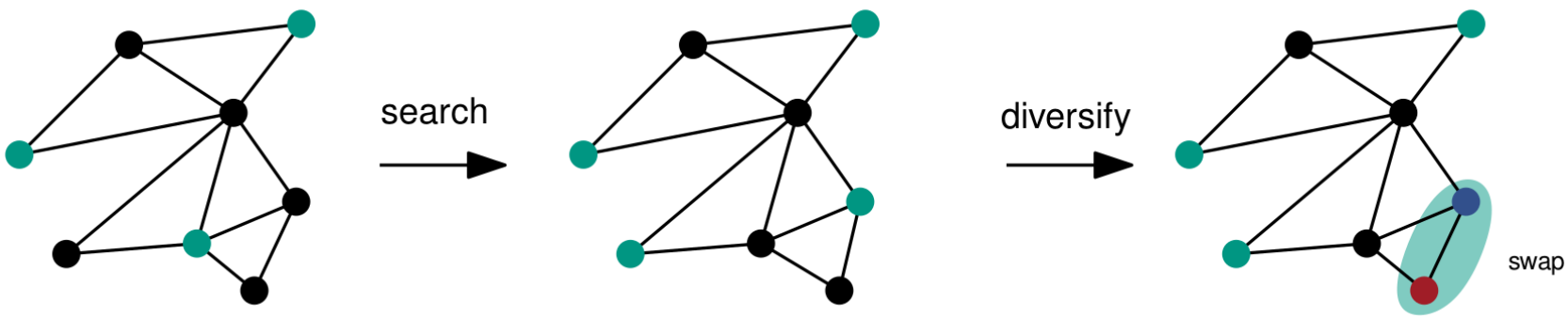
- Many networks with **small kernels** are easy to solve
- Similar instances with **large kernels** remain unsolved

Heuristic Algorithms

Huge real-world networks infeasible for exact algorithms
$\rightarrow$ finding **high-quality** approximations in short time

Recent Approaches:

- Local search based on swaps [Andrade et al. 2012]

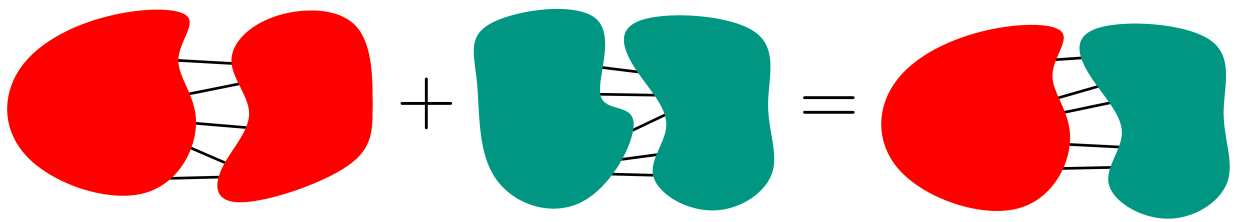


Heuristic Algorithms

Huge real-world networks infeasible for exact algorithms
$\rightarrow$ finding **high-quality** approximations in short time

Recent Approaches:

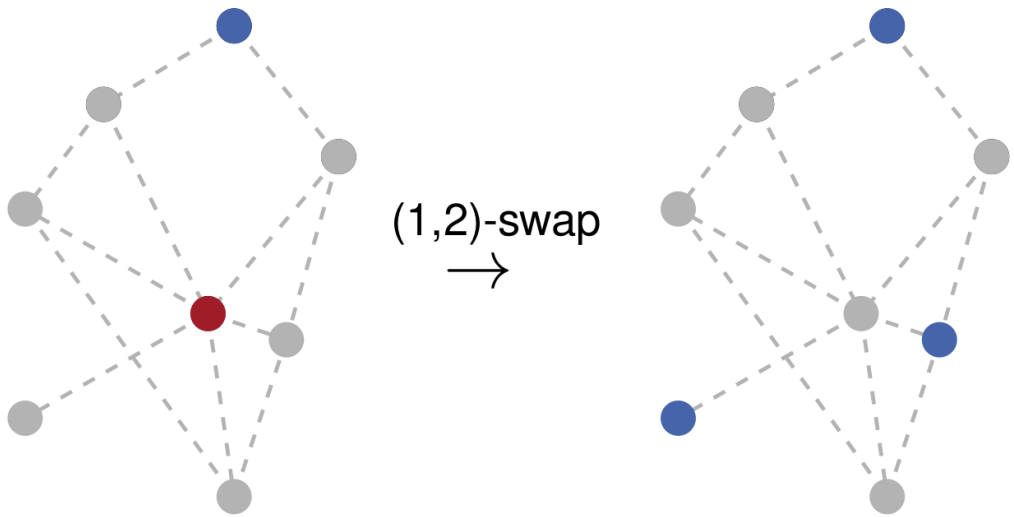
- Local search based on swaps [Andrade et al. 2012]
- Evolutionary algorithms (EvoMIS) [SEA'15]



Iterated Local Search

[Andrade et al.'12]

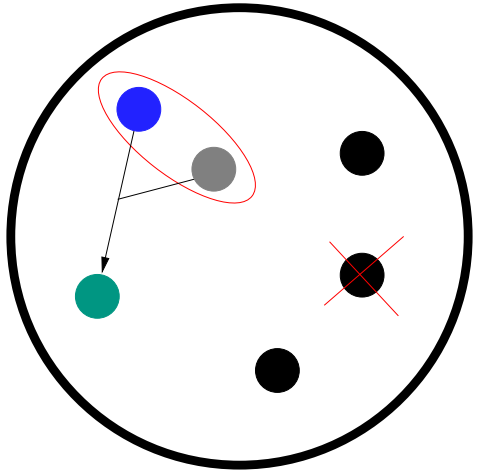
- Compute initial maximal independent set
- Improve using $(1,2)$-swaps:
  - remove single solution node and insert two new ones
  - search for $(1,2)$-swaps in time $O(m)$
- **Tabu mechanism** for "recently" swapped vertices
- No (1,2)-swap $\rightarrow$ **perturbation step**



Evolutionary Algorithm

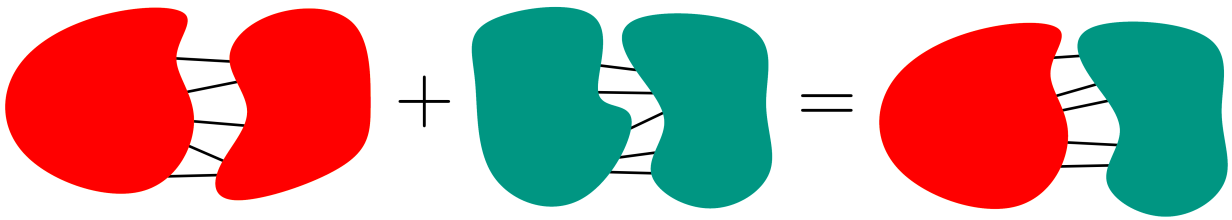
[SEA'15]

create initial population $\mathcal{P}$
while stopping criterion not fulfilled
  select parents $\mathcal{P}_1$, $\mathcal{P}_2$ from $\mathcal{P}$
  recombine $\mathcal{P}_1$ with $\mathcal{P}_2$ to create offspring o
  mutate offspring o
  evict individual in population using o
return *fittest* individual



Recombine Using Graph Partitioning:

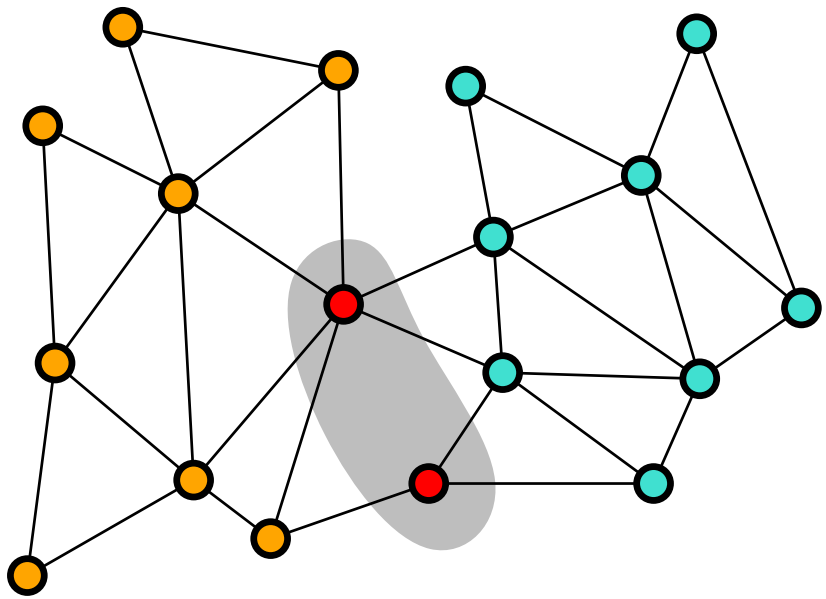
- exchange whole blocks of solutions
- small objective vital for efficiency



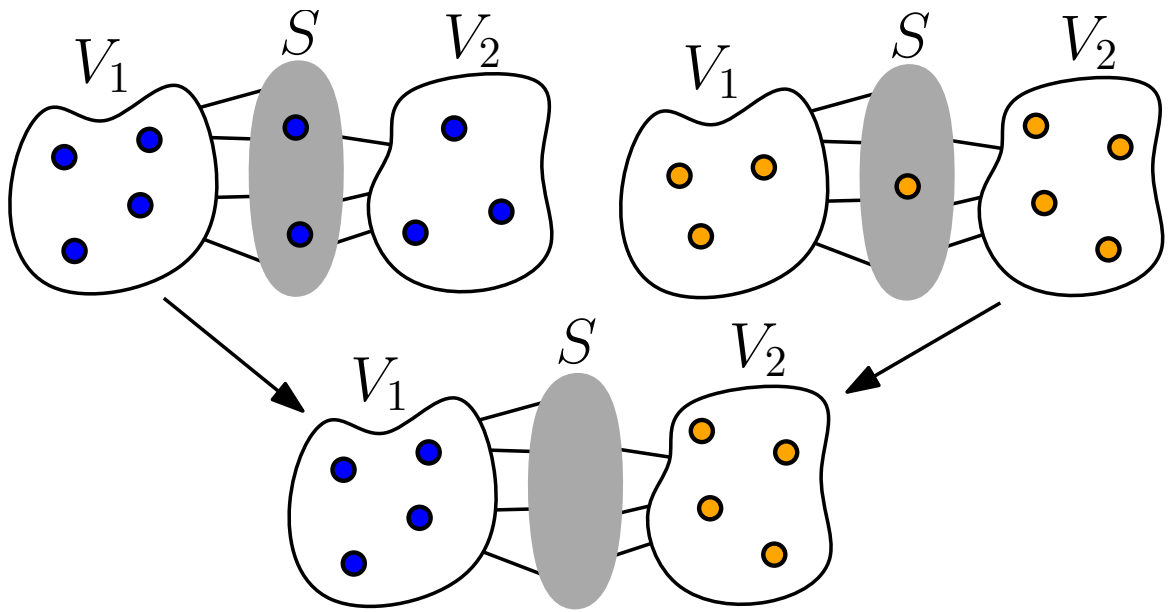
Separators

Partition graph $G = (V, E, c : V \to \mathbf{R}_{>0}, \omega : E \to \mathbf{R}_{>0})$
into k disjoint blocks + node separator s.t.

- total node weight of each block $\leq \frac{1+\epsilon}{k}$ total node weight
- total size of node separator as small as possible

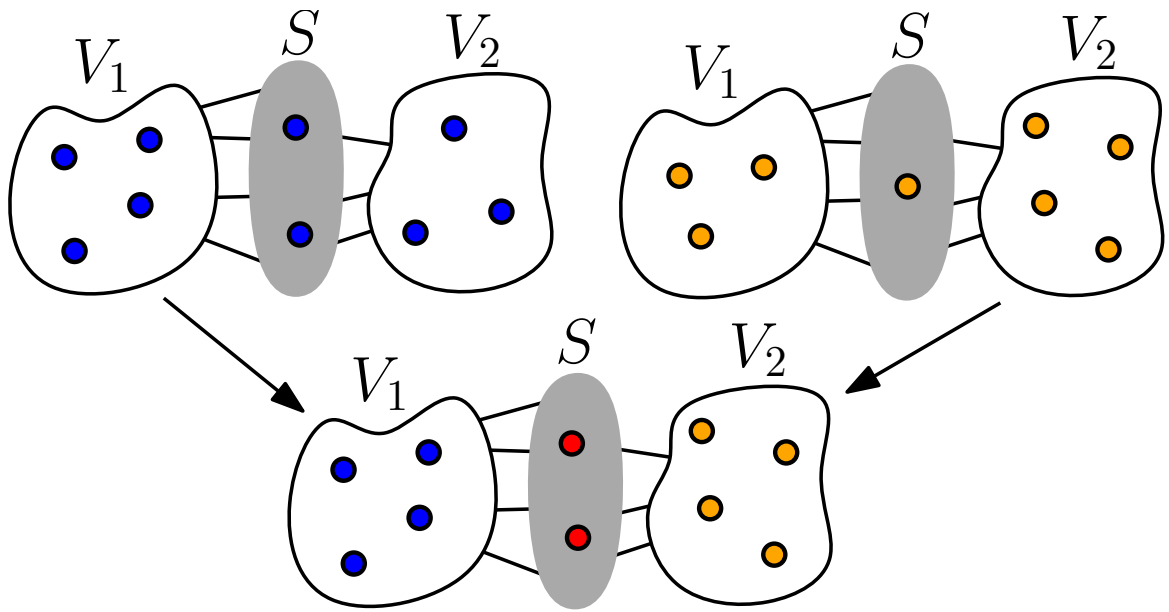


Separator Combine



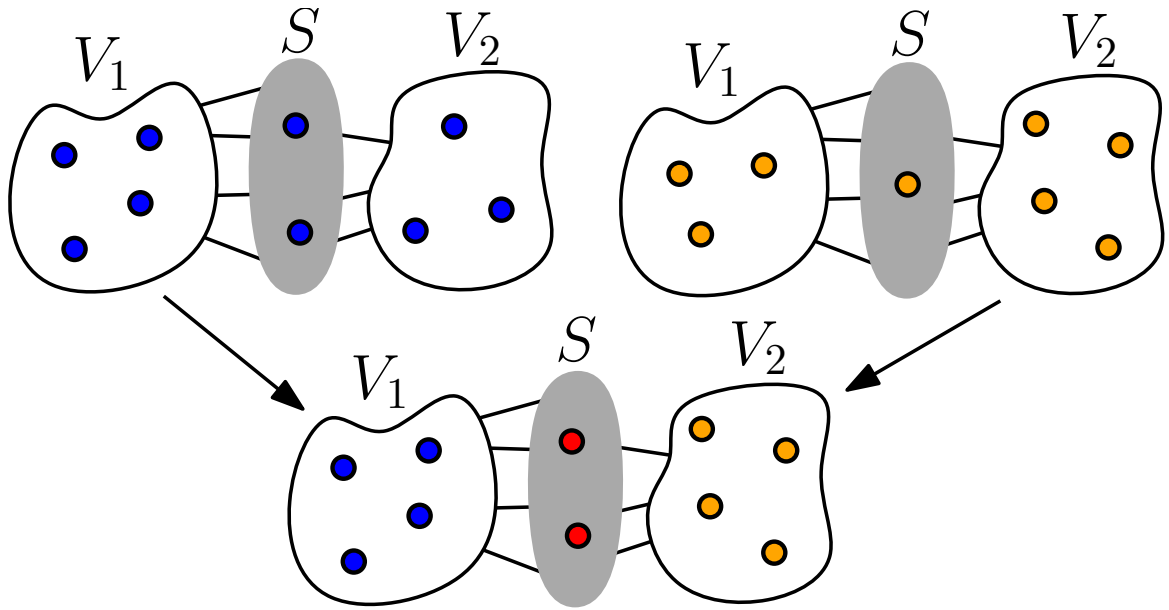
- build node separator $V = V_1 \cup V_2 \cup S$
- use node separator as crossover point
- combination takes linear time $O(n)$
- maximize with local search

Separator Combine



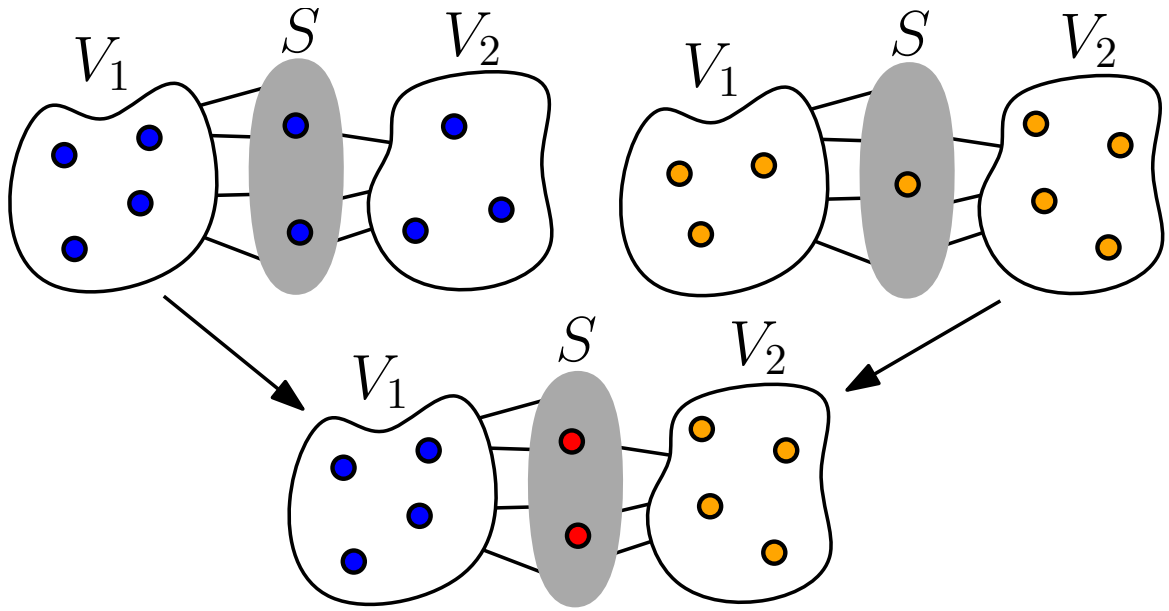
- build node separator $V = V_1 \cup V_2 \cup S$
- use node separator as crossover point
- combination takes linear time $O(n)$
- maximize with local search

Separator Combine



Good Solutions, but Slow

Separator Combine



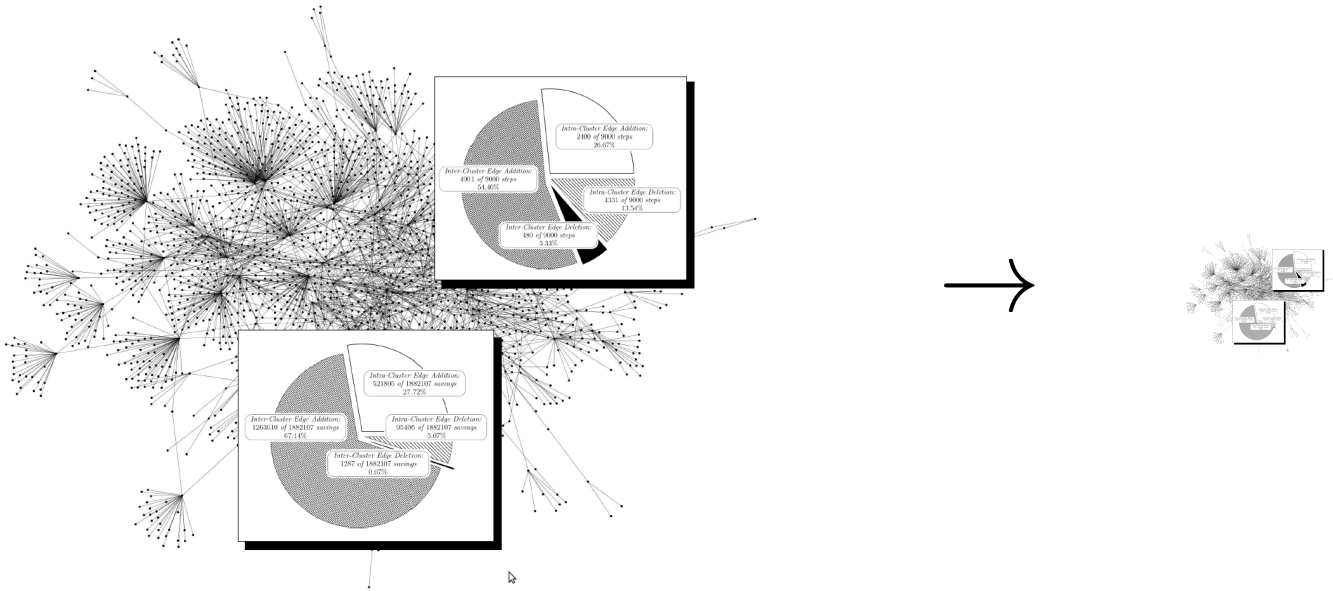
Good Solutions, but Slow

Apply EA on Kernel

Kernelization

[Akiba, Iwata'15]

Reductions:
rules to decrease graph size, while maintaining optimality



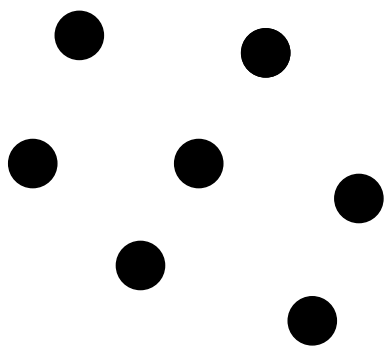
solve problem on problem kernel (using EA)
$\rightarrow$ obtain solution on input graph

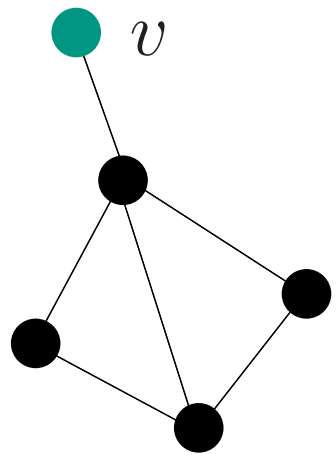
Kernelization

[Akiba, Iwata'15]

Reductions:
rules to decrease graph size, while maintaining optimality

Example:
remove degree 0 or 1 vertices recursively





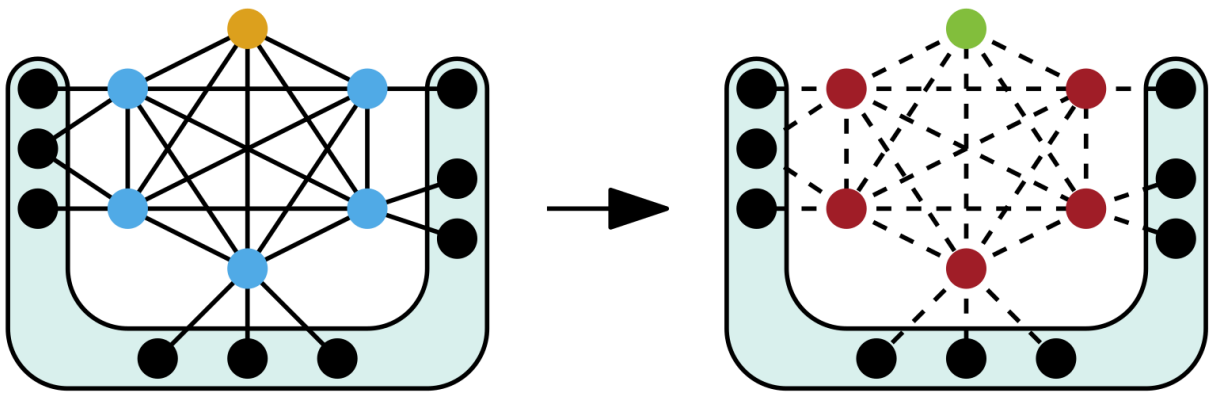
there is always a MIS that contains v
if neighbor of v in MIS choose v instead; otherwise add v

Kernelization

[Akiba, Iwata'15]

Reductions:
rules to decrease graph size, while maintaining optimality

Example:
isolated clique reduction

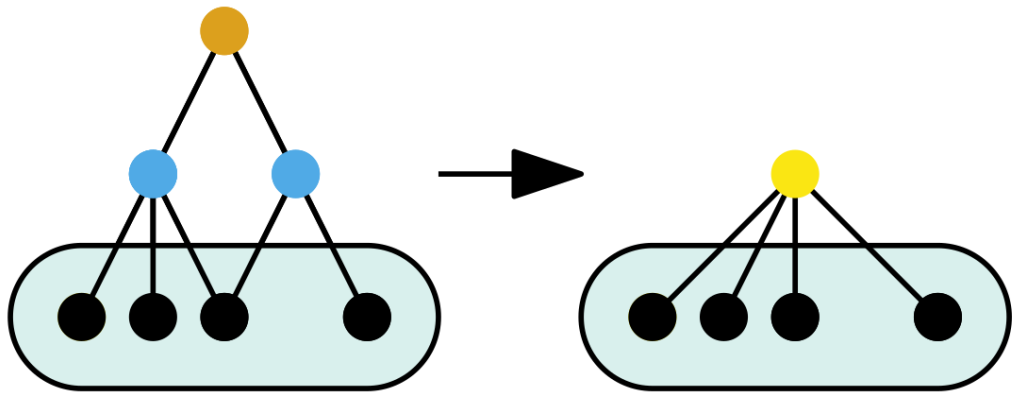


Kernelization

[Akiba, Iwata'15]

Reductions:
rules to decrease graph size, while maintaining optimality

Example:
vertex folding

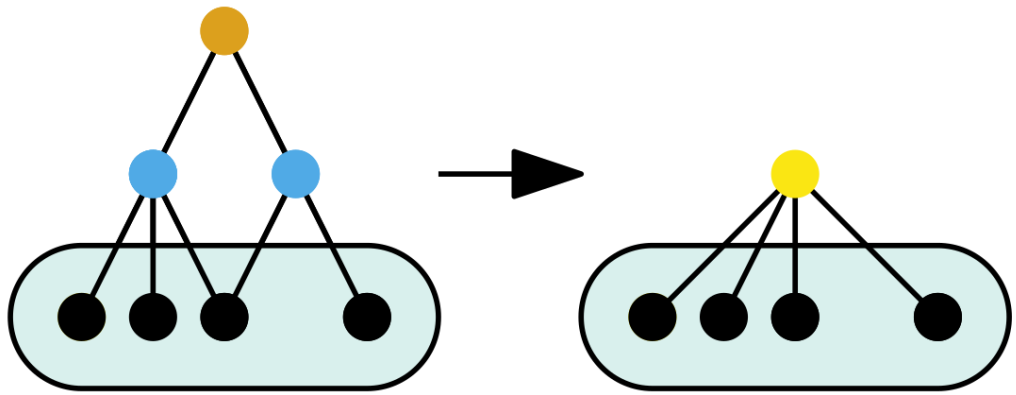


Kernelization

[Akiba, Iwata'15]

Reductions:
rules to decrease graph size, while maintaining optimality

Example:
vertex folding

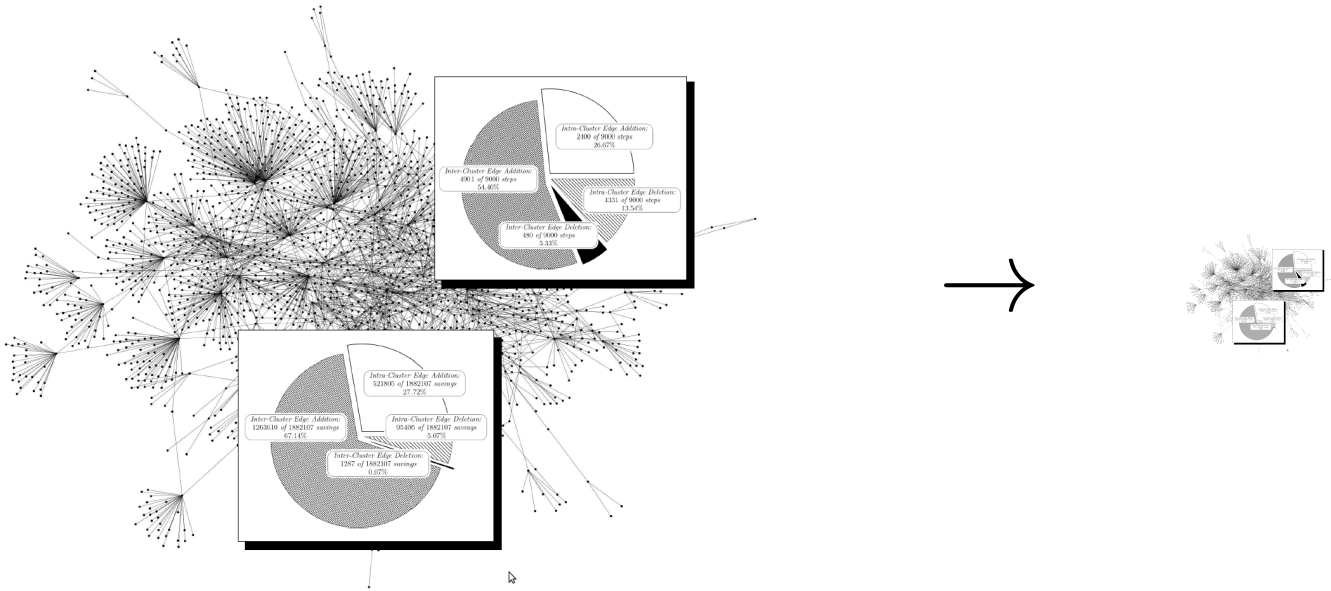


more reductions used in practice → [ALENEX'16]

Kernelization

General Idea

Reductions:
rules to decrease graph size, while maintaining optimality



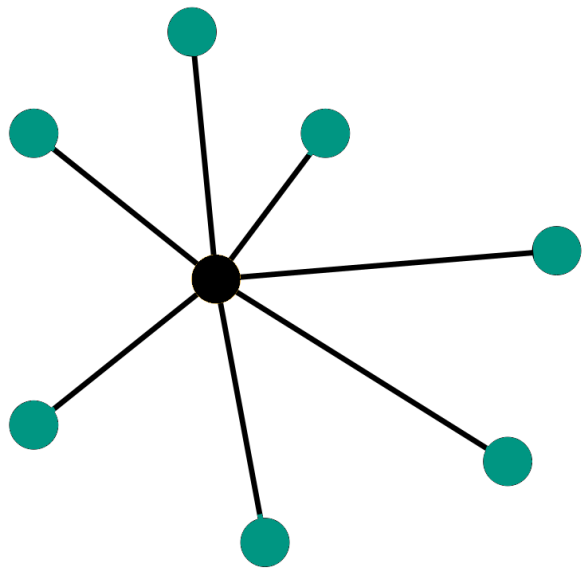
solve problem on problem kernel
$\rightarrow$ obtain solution on input graph quickly

Guess “likely candidates”

can we guess vertices that are in MIS?

idea: select small-degree vertices from “fittest” independent set

apply more reductions and recurse!

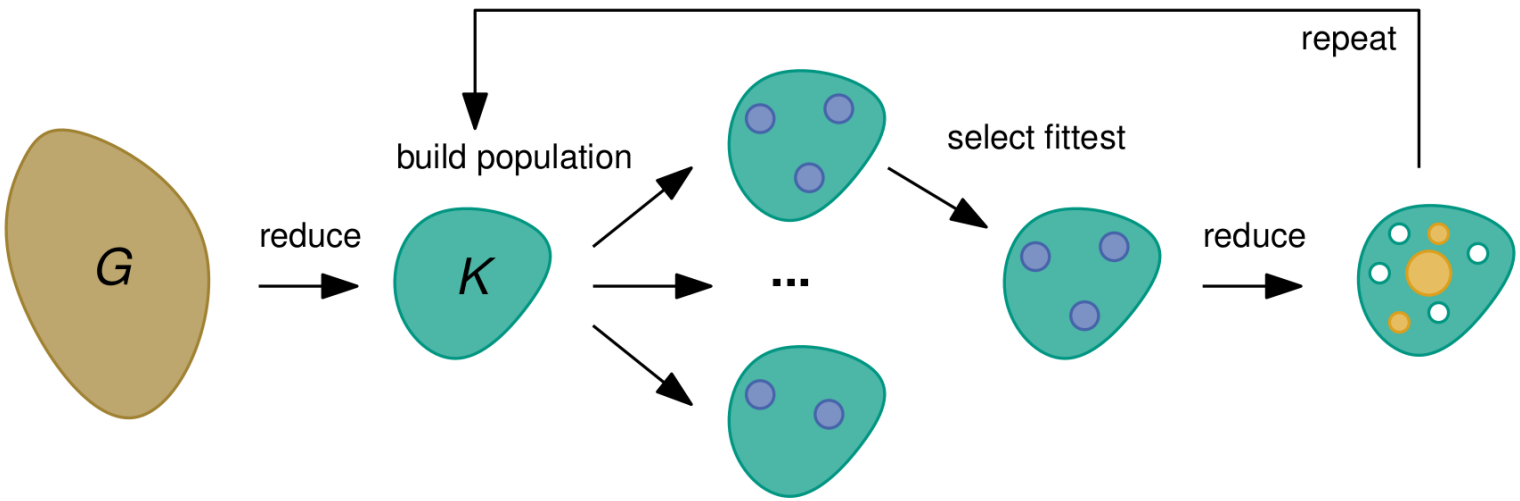


Guess “likely candidates”

can we guess vertices that are in MIS?

idea: select small-degree vertices from “fittest” independent set

apply more reductions and recurse!

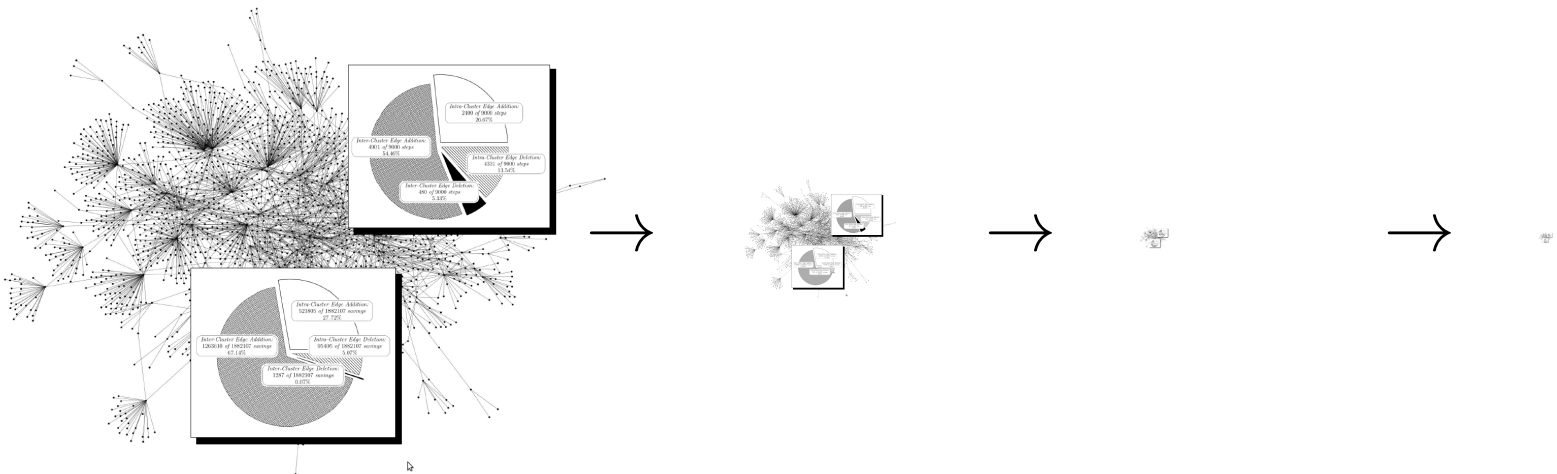


Guess “likely candidates”

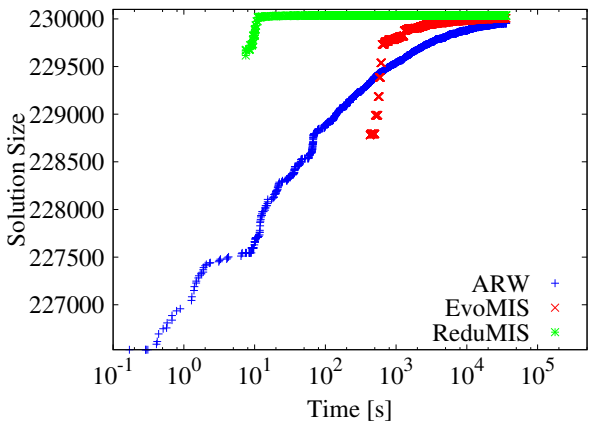
can we guess vertices that are in MIS?

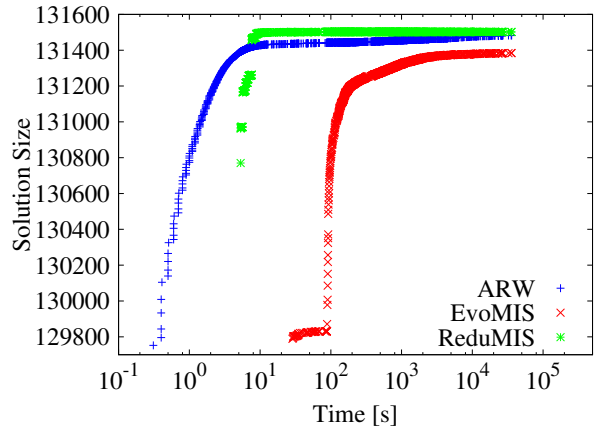
idea: select small-degree vertices from “fittest” independent set

apply more reductions and recurse!



Near-optimal on “difficult networks”

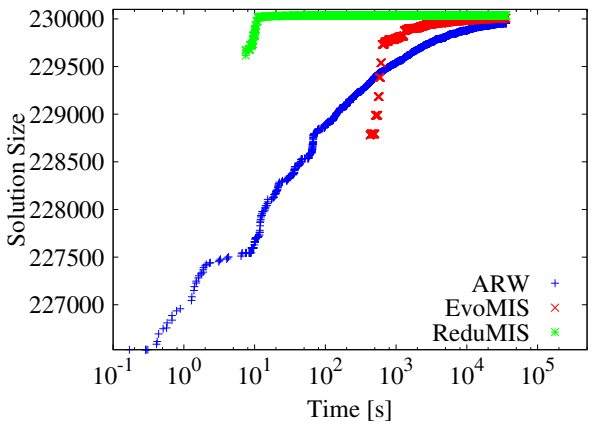


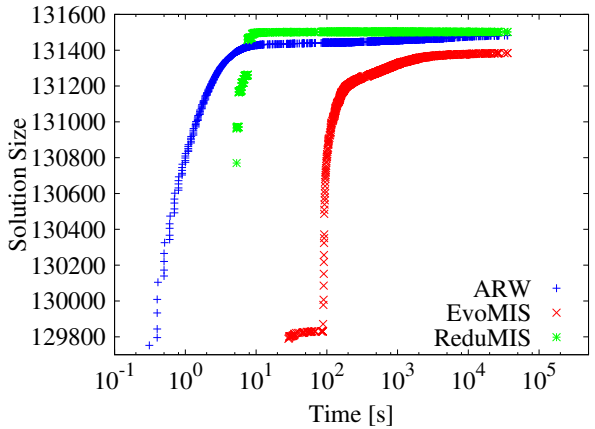


Results:

- finds exact MIS faster, when exact algorithm is slow:
  - Skitter **48 min** $\rightarrow$ **21 min**
  - Stanford **13 hours** $\rightarrow$ **5 min**
  - bcsstk30 **8.6 hours** $\rightarrow$ **2.4 sec**
  - Skitter **2 hours** $\rightarrow$ **28 sec**, ...
- finds exact MIS, for large networks with known MIS size
- consistently finds larger solutions on social and road networks
- even as we scale to graphs to **10M** to **100M** nodes

Near-optimal on “difficult networks”





Problems:

- Kernelization gives **high-quality** at cost of preprocessing time
- ARW is **fast** but struggles with complex scale-free networks

$\rightarrow$ accelerate local search?
$\rightarrow$ parallelization?
$\rightarrow$ get rid of complicated evolutionary algorithm?

ARW on Complex Networks

Issues

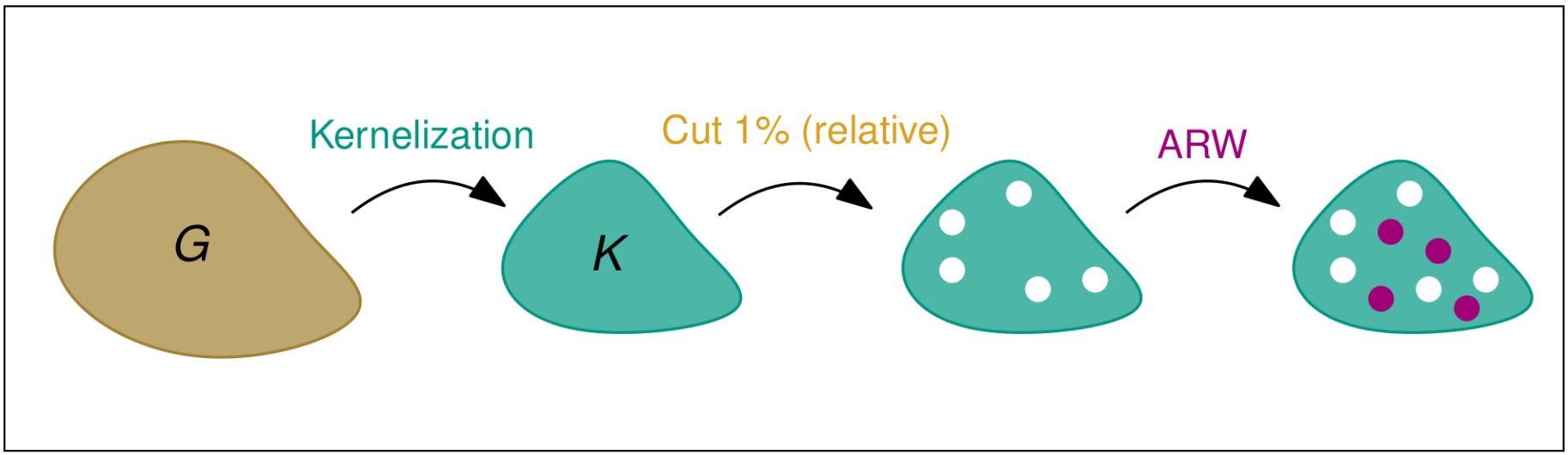
- High degree vertices take long to process ...
  - ... and are unlikely to be in MIS → High-degree cutting
- Many vertices are always in some solution →Kernelization

Strategy I

Full Kernelization

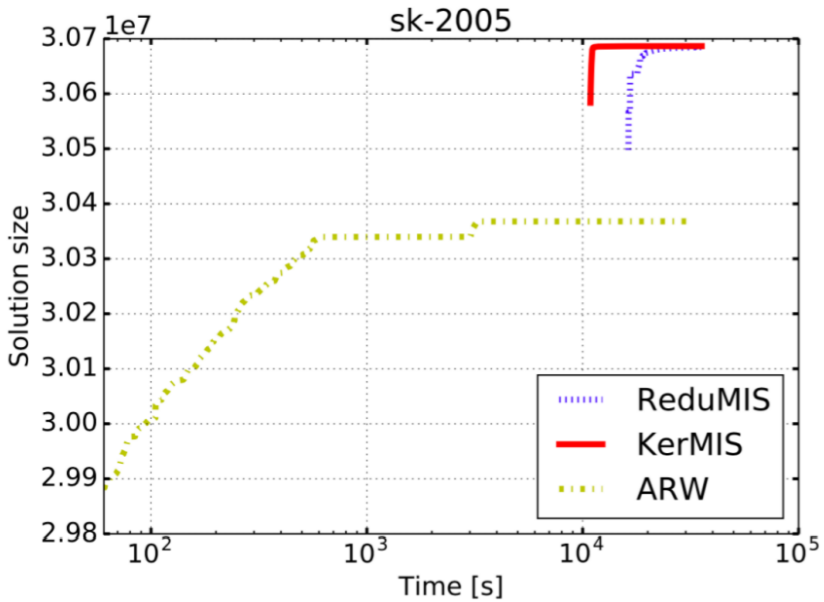
- Repeatedly apply reductions → kernel graph

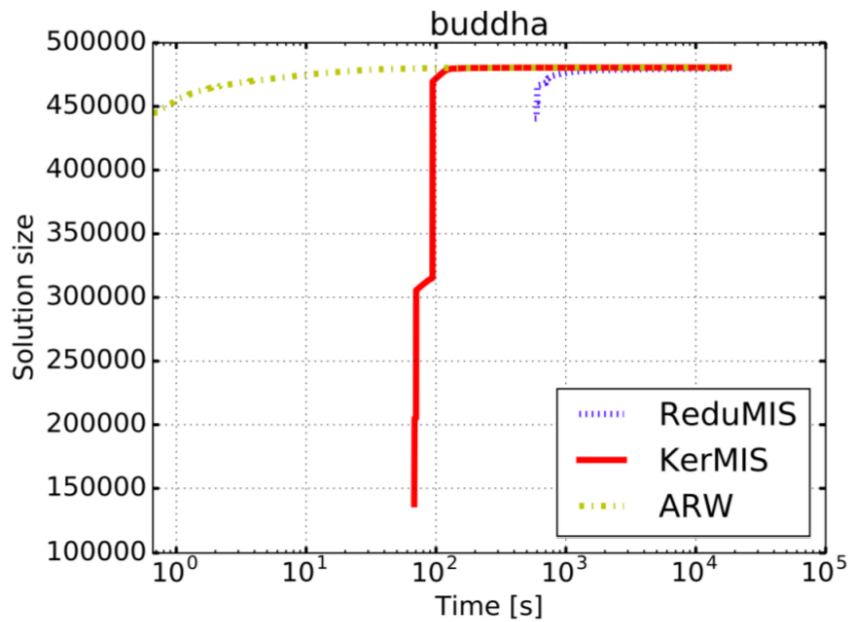
KerMIS:



KerMIS evaluation

- Still long **preprocessing time** to compute kernel
- Inexact reductions **improve performance** of local search
- Quality comparable to **ReduMIS**
  - → Inexact reductions **don't worsen solution quality**

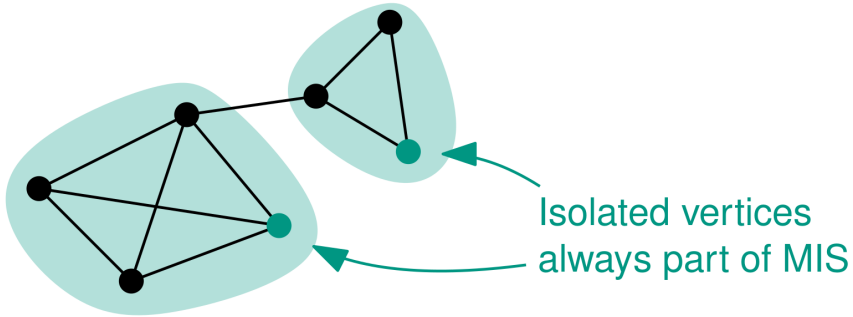




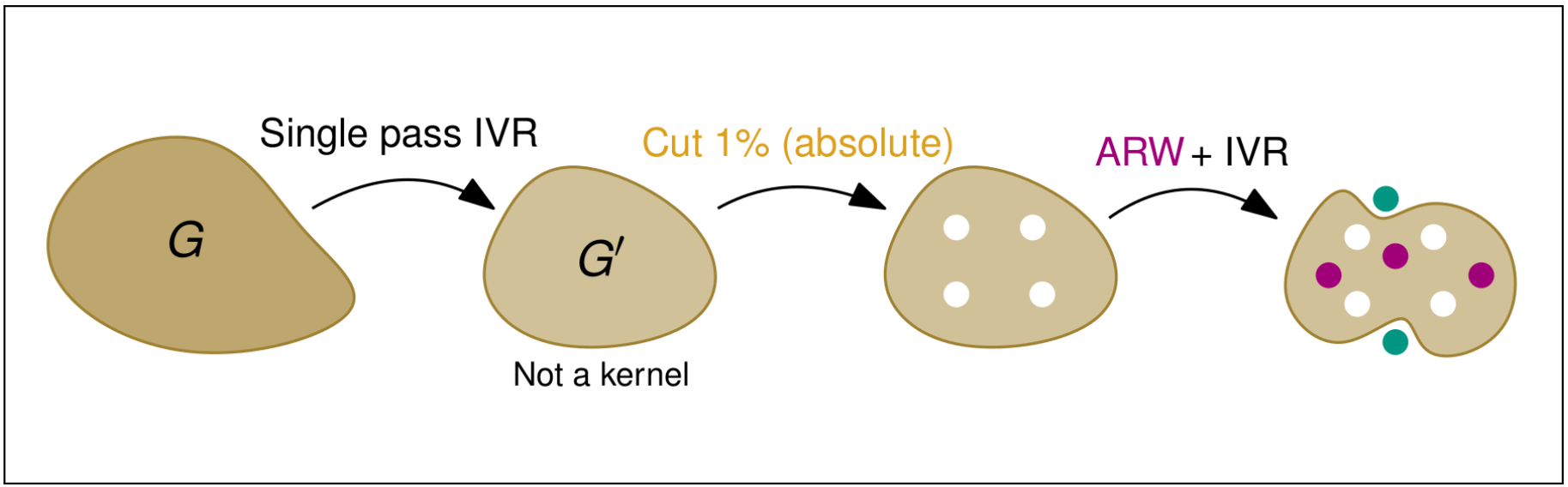
Strategy II

Online "Removal"

- Isolated vertex removal (IVR)

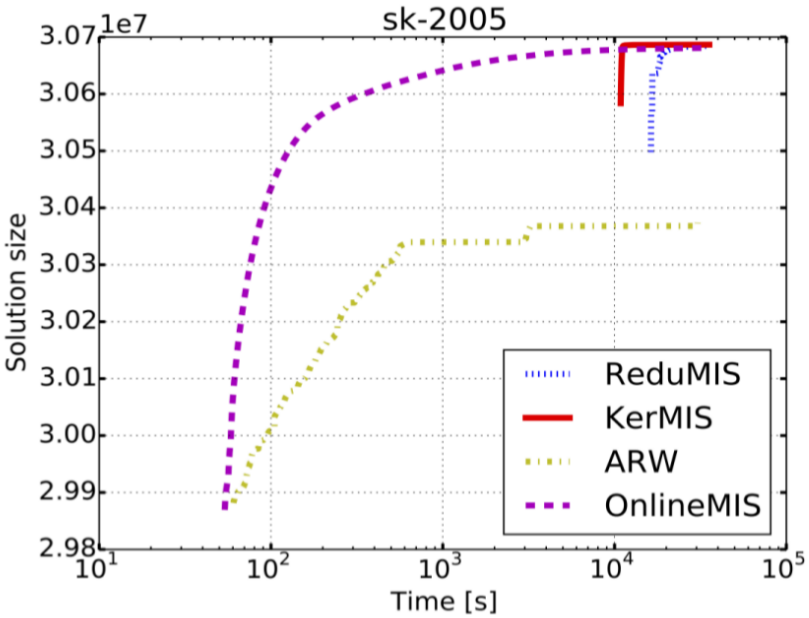


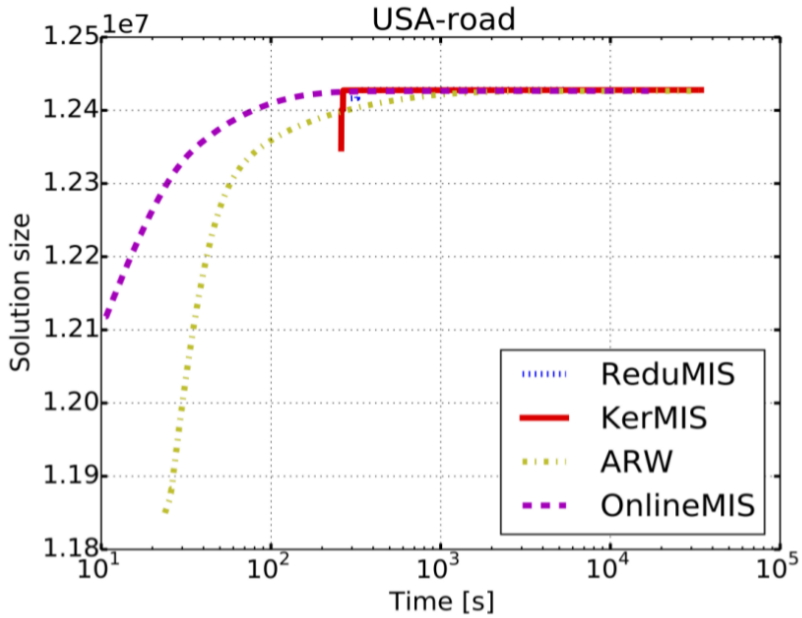
OnlineMIS:



OnlineMIS evaluation

- Significant improvements in **speed and quality**
- **Huge scale-free networks** can be processed efficiently



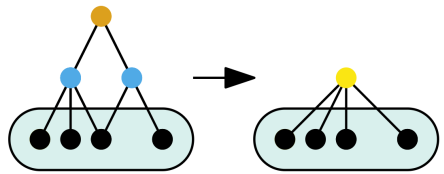


- Maximum speedup $\geq$ 300 over ReduMIS for 14/24 instances

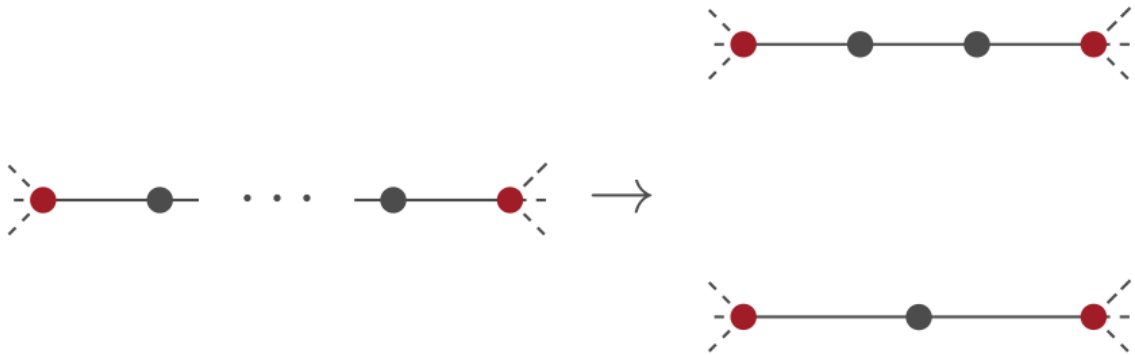
Linear Time Reductions

[Chang et al.'17]

⇝ vertex folding is slow with high-degree neighbors



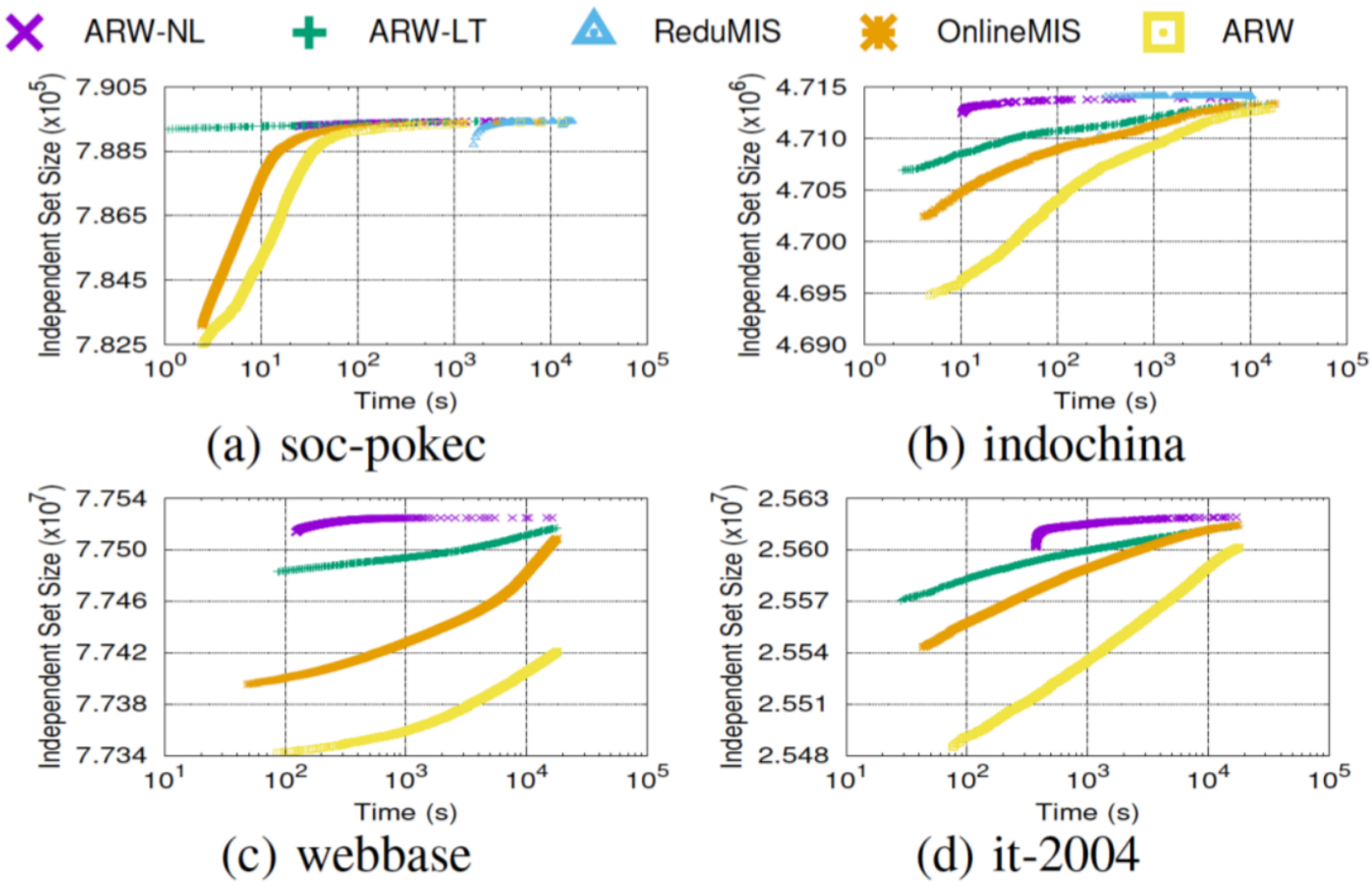
⇝ avoid it



Repeat: add small degree vertices to solution + reduce

Linear Time Reductions

[Chang et al.'17]



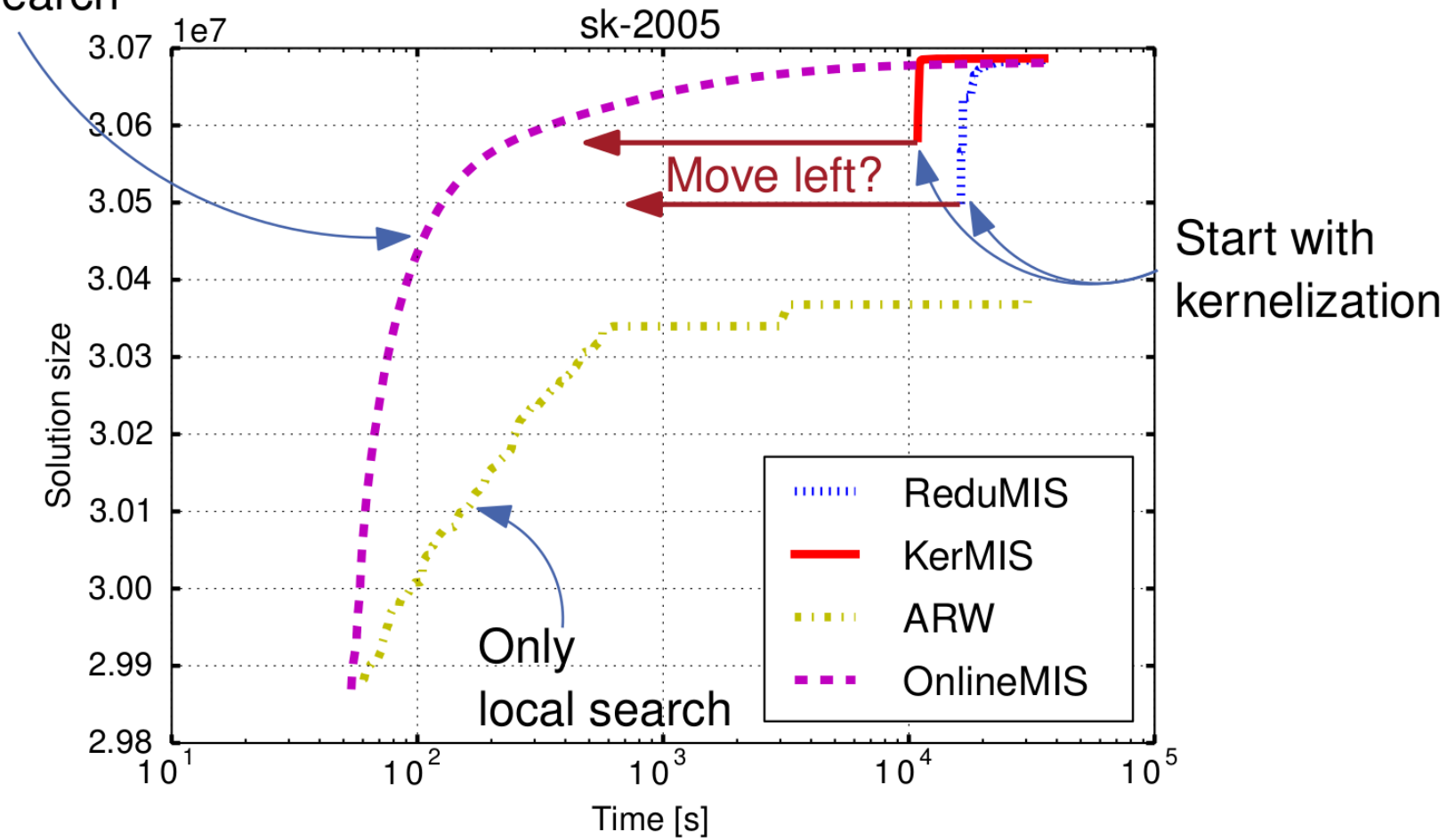
Scalable Reductions

Effective Reductions Are Slow!

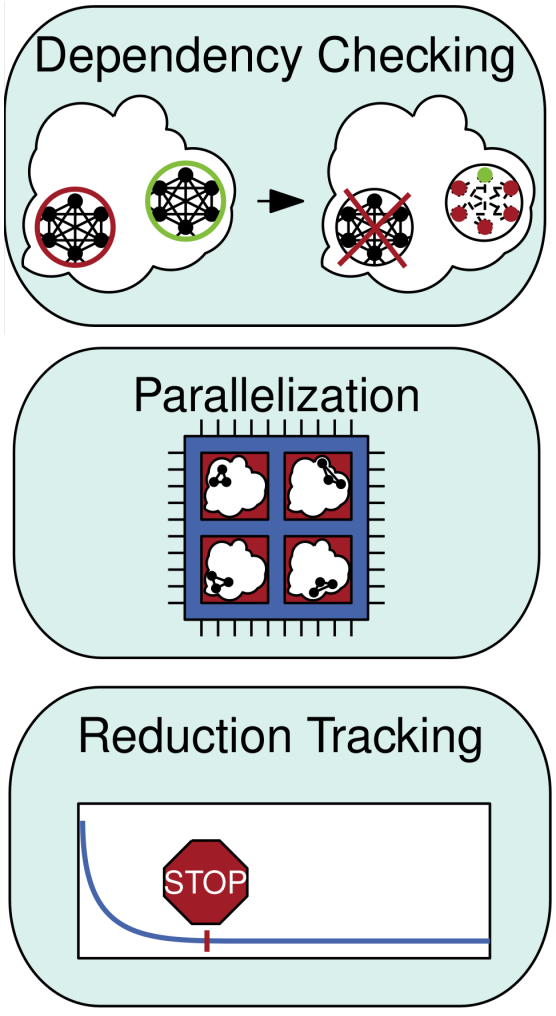
Graph		LinearTime		NearLinear		VCSolver	
name	n	$\|\mathcal{K}\|$	time	$\|\mathcal{K}\|$	time	$\|\mathcal{K}\|$	time
uk-2002	19M	11.7M	1.5	4.0M	28.0	0.2M	336.9
arabic-2005	23M	15.6M	2.6	6.7M	246.1	0.6M	1 033.2
gsh-2015-tpd	31M	2.0M	11.6	1.2M	97.4	0.4M	372.3
uk-2005	39M	28.2M	2.5	5.9M	60.5	0.8M	541.4
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

Even Faster??

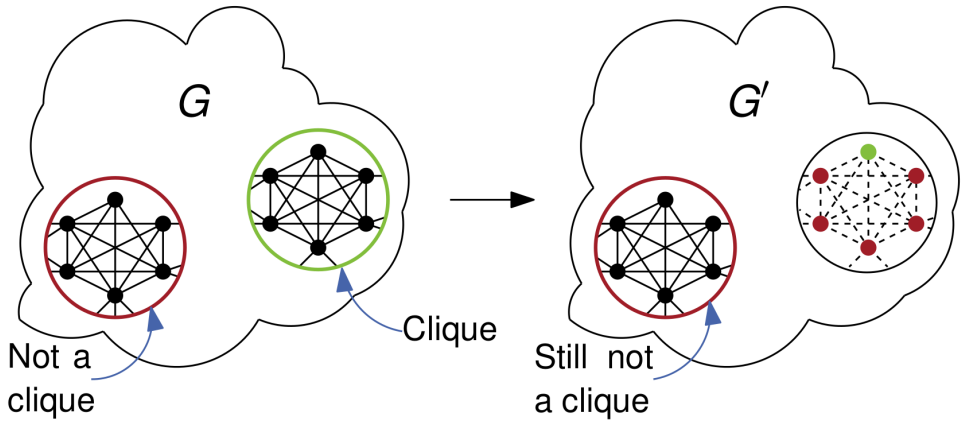
Reductions during local search



Remaining Ingredients



Dependency Checking

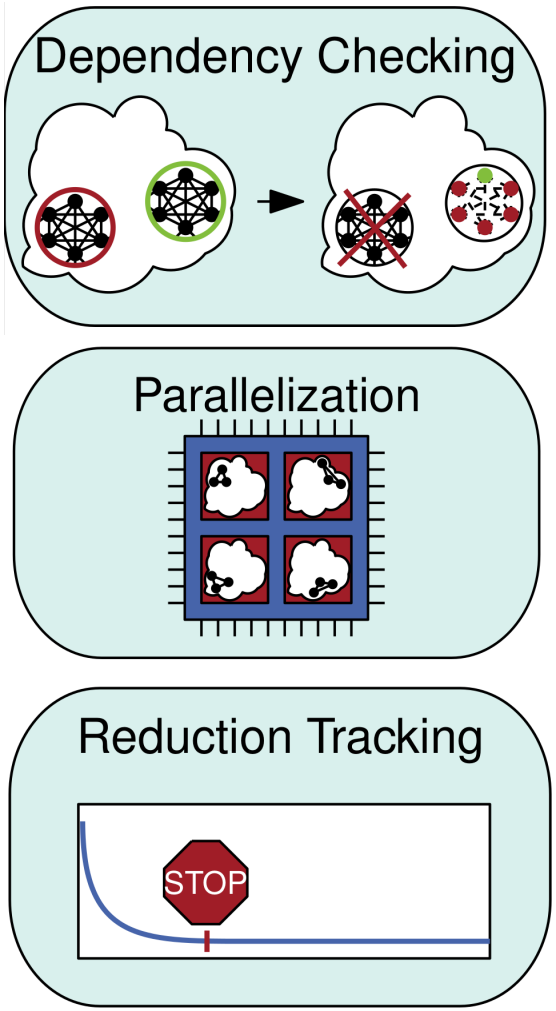


No reduction in G and $N_G(v) = N_{G'}(v) \Rightarrow$ No reduction in G'

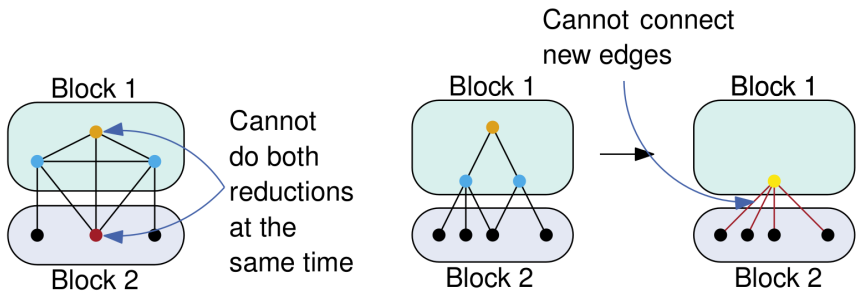
Used for

- Isolated Clique Reduction
- Degree 2 Fold
- Twin Reduction

Remaining Ingredients



Parallelization
partition graph in blocks, reduce separately

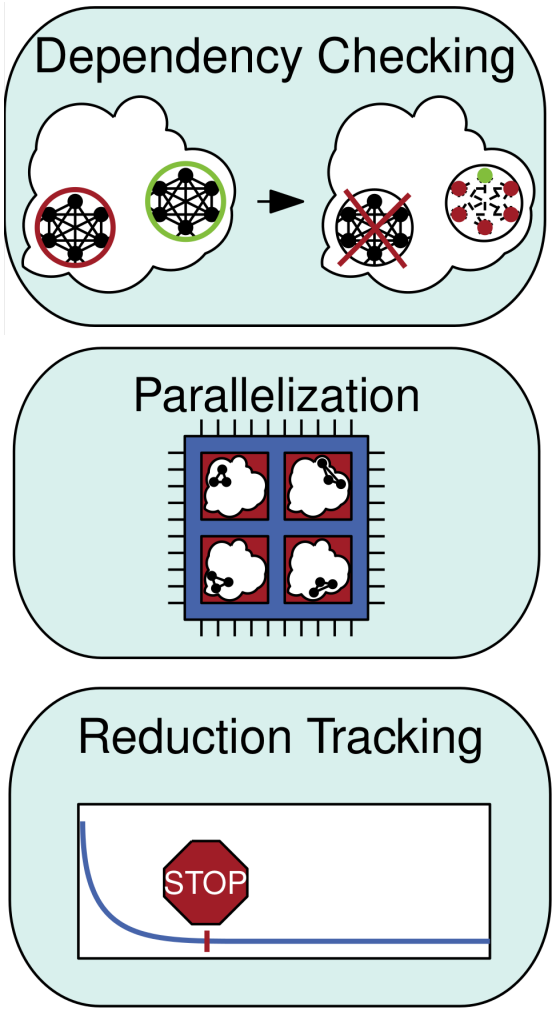


→ deal with conflicts

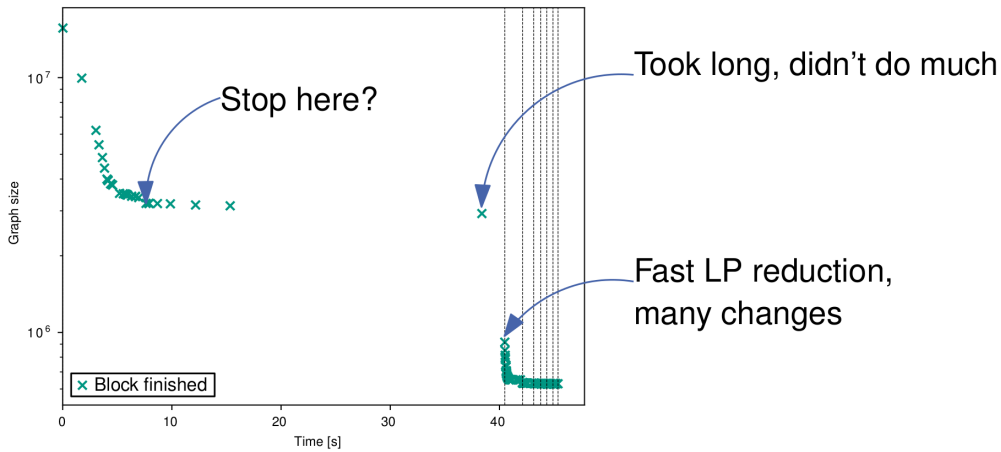
Ingredients:

- ParHIP for partitioning
- Parallelize LP reduction with parallel maximum bipartite matching [Azad et al.'17]

Remaining Ingredients



Reduction Tracking
stop kernelization early



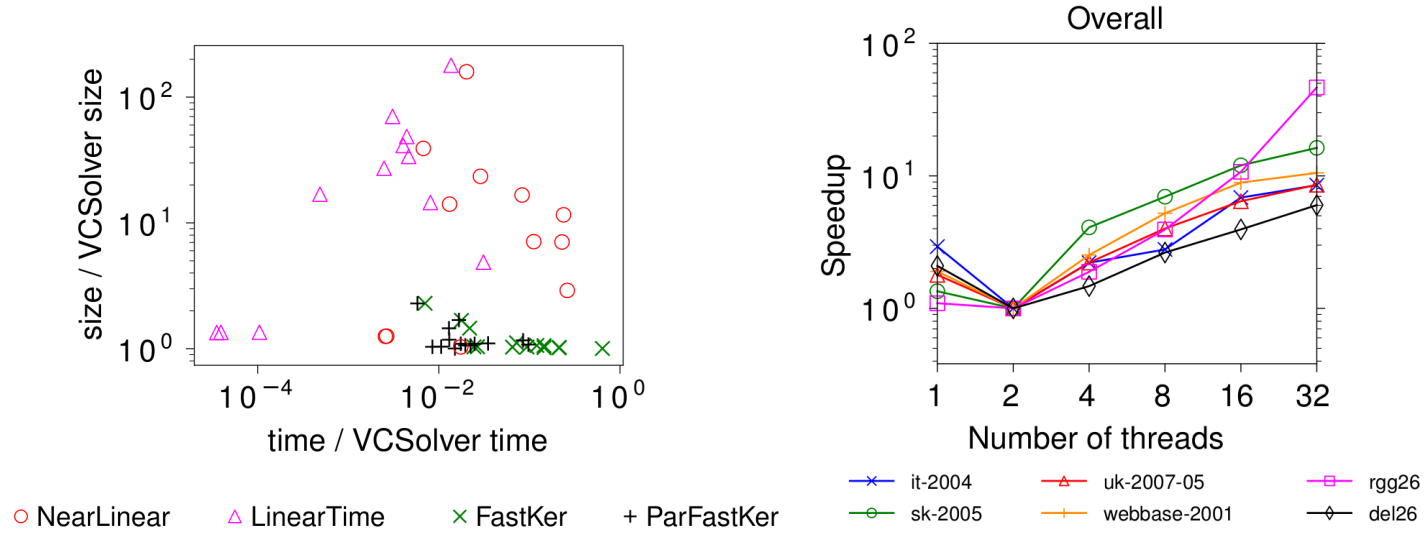
- some blocks take longer than others
- few changes after a while

→ stop "local" reductions early
→ run LP reduction

Scalable Reductions

NearLinear		VCSolver		ParFastKer		
$\|\mathcal{K}\|$	time	$\|\mathcal{K}\|$	time	$\|\mathcal{K}\|$	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	**9.8M**	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

Experimental Results



- VCSolver [Akiba and Iwata'16]: slow but small kernels
- LinearTime and NearLinear [Chang et al.'17]: fast but large kernels
  → integrate as preprocessing

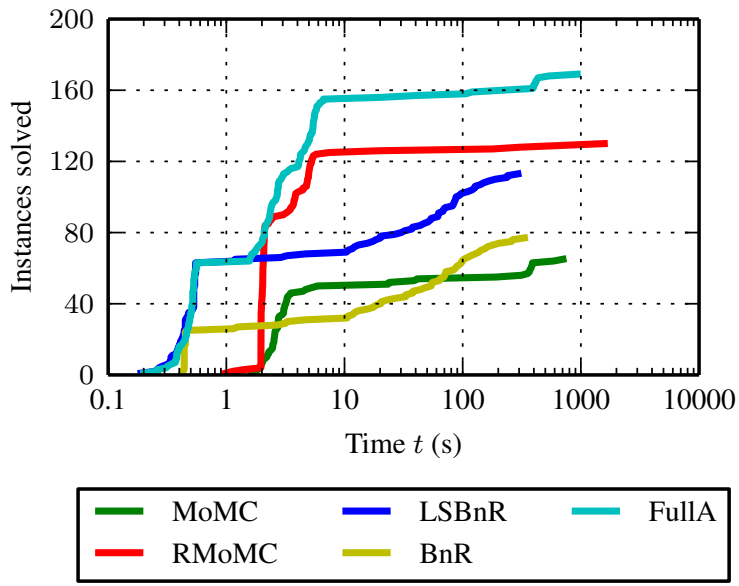
Summary:

- Orders of magnitude smaller than fast methods
- Orders of magnitude faster than algorithm with similar-sized kernels
- Integration into local search (small kernels matter):
  - larger independent sets faster

PACE Challenge 2019

Back to Exact Algorithms ...

- Exact vertex cover solver
  $\rightarrow$based on reductions + local search + portfolio of exact solvers
- **Exact Solvers:**
  - Branch-and-Reduce for vertex cover
  - Clique solver by Li et al.
  - Get the easy instances fast, then get the hard instances.
- 24 competing algorithms
- We *won*



PACE Challenge 2019

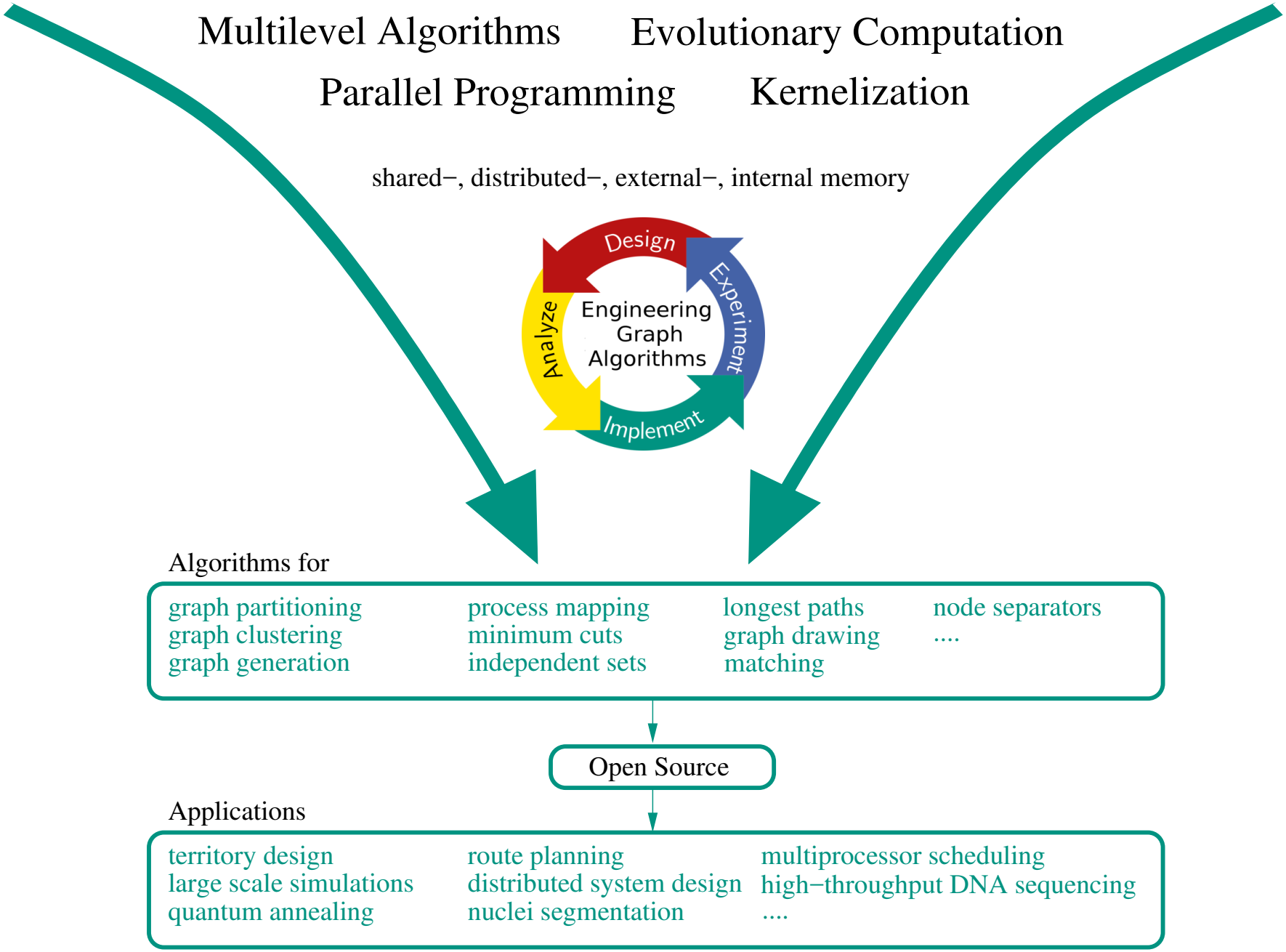
Back to Exact Algorithms ...

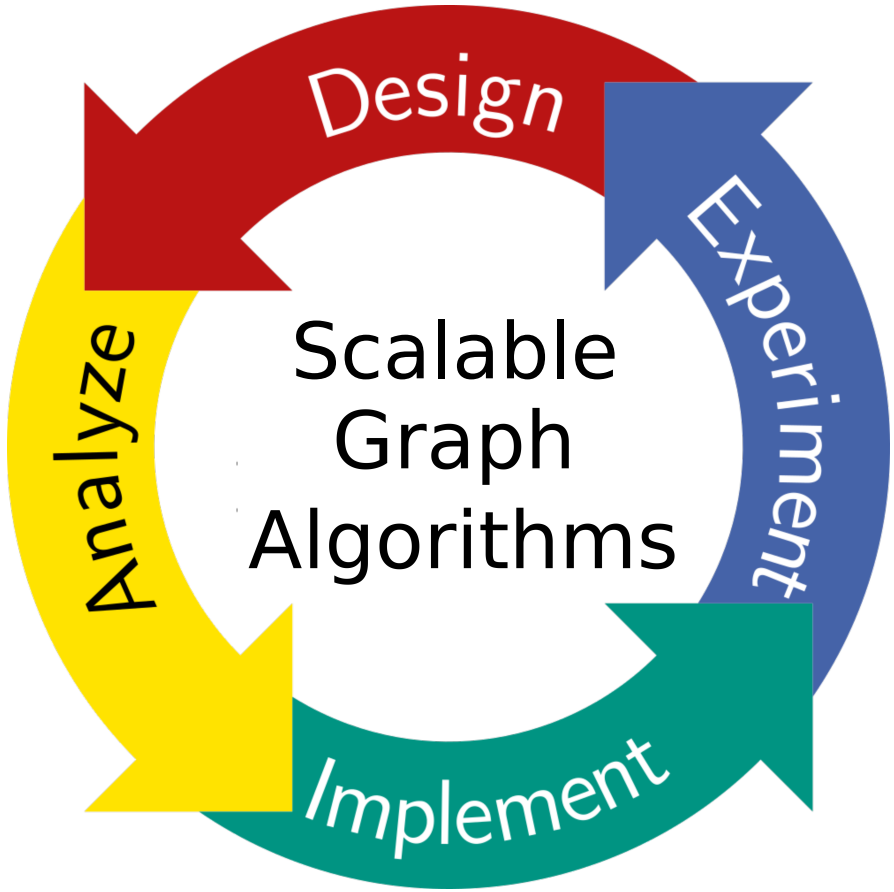
OPTIL.io ABOUT PROBLEMS GLSL TUTORIAL NEWS

Please note that displayed time is user time on CPU. You may receive **TLE** by exceeding limit in real time, while your program's user time is 0, i.e. sleep(10000000)

First 1 Last

#	USER	LANGUAGE	SCORE	TIME [S]	1	2	3	4	5	6	7	8	9	10	11	12	13
1	WeGotYouCovered	Static binary	**84.59**	31,968.68	0.40	**0.00**	1.82	0.15	1.56	0.55	0.08	TLE	0.30	0.41	0.07	0.03	0.0
2	peaty	CMake	**54.60**	43,365.30	0.18	**0.00**	**0.24**	0.12	**0.30**	0.20	**0.00**	TLE	0.14	0.13	0.05	0.11	0.0
3	vasily.alferov	C++	**30.85**	79,474.67	0.10	**0.00**	0.69	0.07	2.06	0.06	**0.00**	TLE	0.10	0.15	4.10	0.13	0.0
4	ksimonov	CMake	**29.90**	119,019.56	0.08	**0.00**	2.58	0.05	4.22	0.12	0.02	TLE	**0.01**	0.10	0.02	0.04	0.0
5	opm	Static binary	**29.58**	122,553.49	0.38	0.12	1.39	0.28	4.88	0.28	0.08	TLE	0.22	0.29	0.18	0.23	0.1
6	tob	Static binary	**29.39**	91,191.78	3.97	0.09	2.27	0.04	3.13	6.81	0.96	MLE	2.87	6.30	0.20	0.14	0.0
7	hub	Static binary	**28.96**	124,576.85	**0.00**	**0.00**	1.58	0.04	3.58	**0.04**	0.03	TLE	0.06	**0.08**	0.02	0.02	0.0
8	sfs	CMake	**28.29**	119,732.27	2.71	**0.00**	2.77	0.11	7.00	2.84	0.72	TLE	2.55	3.06	**0.00**	**0.00**	**0.0**
9	bogdan	Static binary	**28.13**	63,740.73	1.72	0.06	28.47	0.78	29.94	1.50	0.53	TLE	1.48	1.54	0.64	0.46	0.2
10	Vertex_Cover_Solver	C	**23.71**	128,498.75	2.50	0.14	2.08	0.24	6.63	3.02	0.62	TLE	2.28	3.02	2.69	1.70	0.1
11	pace.challenge	Static binary	**21.71**	14,694.65	192.37	127.98	122.44	102.44	97.88	174.41	138.37	**131.51**	117.72	186.72	186.40	120.58	120.3
12	swats	Static binary	**18.22**	15,075.24	75.35	0.16	124.57	7.99	138.93	88.73	81.62	143.08	88.90	78.46	WA	WA	WA
13	xelmh	CMake	**15.02**	143,878.07	0.12	0.05	26.87	0.44	85.23	0.15	0.02	TLE	0.14	0.13	2.51	5.92	0.1
14	pptale	CMake	**11.93**	157,640.11	1.07	**0.00**	1,074.90	**0.03**	TLE	1.10	0.39	TLE	1.49	1.94	1.74	**0.00**	0.0
15	vader95	Python3	**11.71**	154,823.74	4.69	2.00	629.04	4.87	832.33	4.86	2.59	TLE	4.16	5.03	2.40	50.80	2.4
16	arrighi	CMake	**11.56**	158,468.48	8.19	0.18	TLE	0.17	TLE	10.08	2.62	TLE	7.82	9.33	12.00	0.14	0.1
17	johannes	CMake	**10.98**	154,533.16	0.23	0.10	WA	0.32	WA	0.27	0.08	TLE	0.22	0.26	TLE	0.46	0.1
18	tomglint	Static binary	**7.99**	162,464.67	0.14	**0.00**	TLE	TLE	TLE	0.18	**0.00**	TLE	0.16	0.17	TLE	TLE	TLE
19	NTU	C++	**3.91**	8,253.19	MLE	0.32	WA	0.88	WA	MLE	9.63	WA	MLE	MLE	WA	1.10	WA


Multilevel Algorithms
Evolutionary Computation
Parallel Programming
Kernelization
shared−, distributed−, external−, internal memory
Design
Experiment
Implement
Analyze
Engineering Graph Algorithms
Algorithms for
graph partitioning
graph clustering
graph generation
process mapping
minimum cuts
independent sets
longest paths
graph drawing
matching
node separators
....
Open Source
Applications
territory design
large scale simulations
quantum annealing
route planning
distributed system design
nuclei segmentation
multiprocessor scheduling
high−throughput DNA sequencing
....


Design
Experiment
Implement
Analyze
Scalable
Graph
Algorithms

Room 6.03