

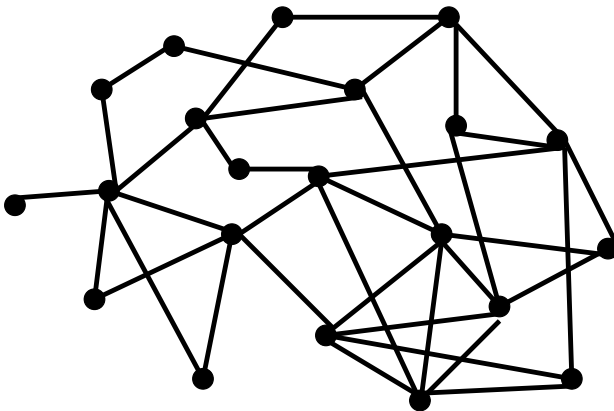
A Stochastic Local Search Approach to Vertex Cover

Silvia Richter, Malte Helmert & Charles Gretton

September 11, 2007

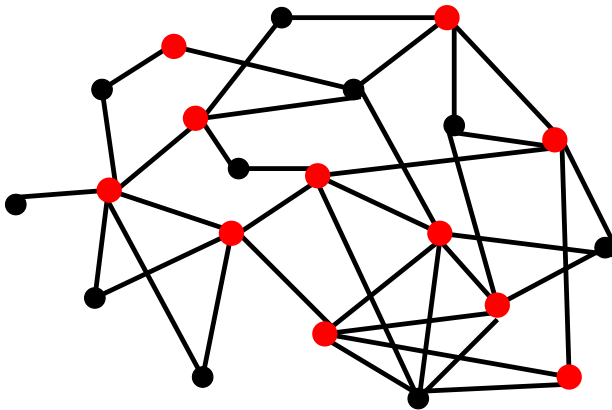
Example

Task: Control all network links using ≤ 11 controllers



Example

Task: Control all network links using ≤ 11 controllers



k -VERTEX COVER

Input: Graph $G = (V, E)$ and positive integer k

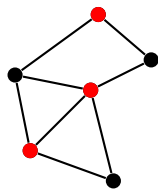
Task: Find $C \subseteq V$ s. t. all edges have an endpoint in C
and $|C| \leq k$

NP-complete

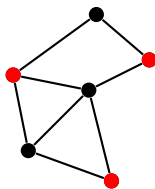
Applications in biology, networks, scheduling, VLSI design,...

Independent Set: subset of vertices that are not connected

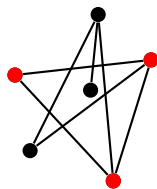
Clique: subset of vertices that are fully connected



vertex cover



independent set



clique
(complement graph)

Stochastic Local Search

Popular approach to NP-hard problems (e. g. SAT)

Begin with some candidate solution S .

While S is not a solution. . .

- Modify S slightly.

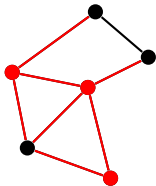
Algorithm Overview

The COVER algorithm:

Begin with a greedily constructed set C .

Iteratively...

- 1 Randomly select an uncovered edge (u_1, u_2) .
- 2 Heuristically choose $u \in \{u_1, u_2\}$ and $v \in C$.
- 3 $C := C - \{v\} \cup \{u\}$
- 4 Test whether C is a vertex cover.



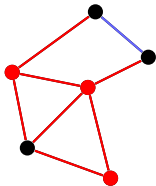
Algorithm Overview

The COVER algorithm:

Begin with a greedily constructed set C .

Iteratively...

- 1 Randomly select an uncovered edge (u_1, u_2) .
- 2 Heuristically choose $u \in \{u_1, u_2\}$ and $v \in C$.
- 3 $C := C - \{v\} \cup \{u\}$
- 4 Test whether C is a vertex cover.



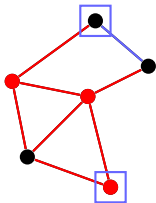
Algorithm Overview

The COVER algorithm:

Begin with a greedily constructed set C .

Iteratively...

- 1 Randomly select an uncovered edge (u_1, u_2) .
- 2 Heuristically choose $u \in \{u_1, u_2\}$ and $v \in C$.
- 3 $C := C - \{v\} \cup \{u\}$
- 4 Test whether C is a vertex cover.



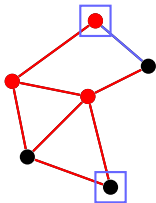
Algorithm Overview

The COVER algorithm:

Begin with a greedily constructed set C .

Iteratively...

- 1 Randomly select an uncovered edge (u_1, u_2) .
- 2 Heuristically choose $u \in \{u_1, u_2\}$ and $v \in C$.
- 3 $C := C - \{v\} \cup \{u\}$
- 4 Test whether C is a vertex cover.



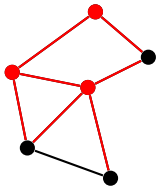
Algorithm Overview

The COVER algorithm:

Begin with a greedily constructed set C .

Iteratively...

- 1 Randomly select an uncovered edge (u_1, u_2) .
- 2 Heuristically choose $u \in \{u_1, u_2\}$ and $v \in C$.
- 3 $C := C - \{v\} \cup \{u\}$
- 4 Test whether C is a vertex cover.



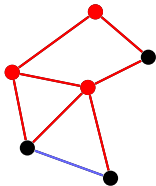
Algorithm Overview

The COVER algorithm:

Begin with a greedily constructed set C .

Iteratively...

- 1 Randomly select an uncovered edge (u_1, u_2) .
- 2 Heuristically choose $u \in \{u_1, u_2\}$ and $v \in C$.
- 3 $C := C - \{v\} \cup \{u\}$
- 4 Test whether C is a vertex cover.



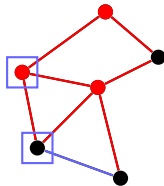
Algorithm Overview

The COVER algorithm:

Begin with a greedily constructed set C .

Iteratively...

- 1 Randomly select an uncovered edge (u_1, u_2) .
- 2 Heuristically choose $u \in \{u_1, u_2\}$ and $v \in C$.
- 3 $C := C - \{v\} \cup \{u\}$
- 4 Test whether C is a vertex cover.



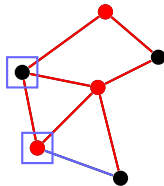
Algorithm Overview

The COVER algorithm:

Begin with a greedily constructed set C .

Iteratively...

- 1 Randomly select an uncovered edge (u_1, u_2) .
- 2 Heuristically choose $u \in \{u_1, u_2\}$ and $v \in C$.
- 3 $C := C - \{v\} \cup \{u\}$
- 4 Test whether C is a vertex cover.

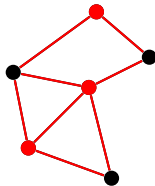


The COVER algorithm:

Begin with a greedily constructed set C .

Iteratively...

- 1 Randomly select an uncovered edge (u_1, u_2) .
- 2 Heuristically choose $u \in \{u_1, u_2\}$ and $v \in C$.
- 3 $C := C - \{v\} \cup \{u\}$
- 4 Test whether C is a vertex cover.



The Search Heuristic I

Edge weights: positive real numbers associated with edges

High weight \Rightarrow “difficult” edge

- initialized to constant
- incremented at each iteration for *uncovered* edges

Vertex weights: derived from weights of those incident edges that a vertex is “responsible” for

Higher weight \Rightarrow vertex more likely to be chosen

Swap vertices a and b to maximise **gain**:

$$weight(b) - weight(a) + \delta$$

Additional conditions:

Taboo list: the 2 vertices last removed/inserted

- Not allowed to be chosen for inclusion/removal in C
- Preserves most recent decision

Time stamp: iteration when a vertex was last removed

- Prefer vertices with lower time stamp
- Used to break ties

The COVER Algorithm

- 1: initialize C greedily with $|C| = k$
- 2: initialize weights
- 3: $iteration_number = 1$
- 4: **while** exists uncovered edge and
 $iteration_number < \text{MAX_ITERATIONS}$ **do**
- 5: choose uncovered edge (u_1, u_2) randomly w. u. p.
- 6: choose vertices $u \in \{u_1, u_2\}$ and $v \in C$ according to
max gain criterion, taboo list & time stamps
- 7: $C = C \setminus \{v\}$
- 8: $C = C \cup \{u\}$
- 9: $taboo_list = \{v, u\}$
- 10: $u.time_stamp = iteration_number$
- 11: increase edge weights
- 12: increment $iteration_number$
- 13: **end while**

The COVER Algorithm

- 1: initialize C greedily with $|C| = k$
- 2: initialize weights
- 3: $iteration_number = 1$
- 4: **while** exists uncovered edge and
 $iteration_number < \text{MAX_ITERATIONS}$ **do**
- 5: choose uncovered edge (u_1, u_2) randomly **Exploration**
- 6: choose vertices $u \in \{u_1, u_2\}$ and $v \in C$ according to
max gain criterion, taboo list & time stamps **Exploitation**
- 7: $C = C \setminus \{v\}$
- 8: $C = C \cup \{u\}$
- 9: $taboo_list = \{v, u\}$
- 10: $u.time_stamp = iteration_number$
- 11: increase edge weights
- 12: increment $iteration_number$
- 13: **end while**

Three benchmark sets:

- Biological problems
- BHOSLIB
- DIMACS

Comparison against different approaches from literature

100 runs with different random seeds

Repeat unsuccessful runs, increasing k

Finding correlated protein sequences

- Largest possible set of correlated sequences
⇒ maximum clique problem
- Problem instances from Abu-Khzam et al. (2006)
- Comparison against Abu-Khzam et al.'s parallel exhaustive search based on *Fixed-Parameter Tractability* (P-FPT)

Experimental Results I

Instance	Graph			COVER		P-FPT Runtime
	$ V $	$ E $	k^*	Quality	Median	
globin3	972	3898	165	100-0-0	0.01	23
globin7	972	38557	350	100-0-0	0.01	47
globin9	972	62525	378	100-0-0	0.01	227
globin15	972	149473	427	98-1-1	0.01	14
sh2-3	839	5860	246	100-0-0	0.01	22
sh2-4	839	13799	337	100-0-0	0.01	2593
sh2-5	839	26612	399	100-0-0	0.01	7
sh2-10	839	129697	547	100-0-0	0.01	332
sh3-10	2466	1508850	2044	100-0-0	0.80	8400

Experimental Results I

Instance	Graph			COVER		P-FPT Runtime
	$ V $	$ E $	k^*	Quality	Median	
globin3	972	3898	165	100-0-0	0.01	23
globin7	972	38557	350	100-0-0	0.01	47
globin9	972	62525	378	100-0-0	0.01	227
globin15	972	149473	427	98-1-1	0.01	14
sh2-3	839	5860	246	100-0-0	0.01	22
sh2-4	839	13799	337	100-0-0	0.01	2593
sh2-5	839	26612	399	100-0-0	0.01	7
sh2-10	839	129697	547	100-0-0	0.01	332
sh3-10	2466	1508850	2044	100-0-0	0.80	8400

Experimental Results I

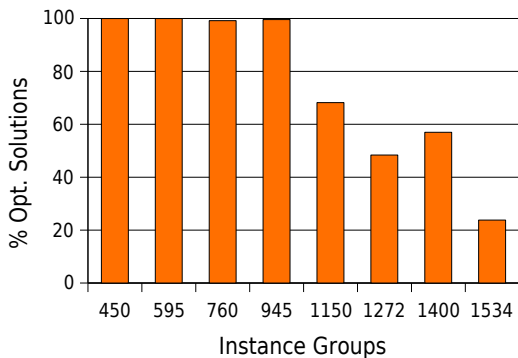
Instance	Graph			COVER		P-FPT
	$ V $	$ E $	k^*	Quality	Median	Runtime
globin3	972	3898	165	100-0-0	0.01	23
globin7	972	38557	350	100-0-0	0.01	47
globin9	972	62525	378	100-0-0	0.01	227
globin15	972	149473	427	98-1-1	0.01	14
sh2-3	839	5860	246	100-0-0	0.01	22
sh2-4	839	13799	337	100-0-0	0.01	2593
sh2-5	839	26612	399	100-0-0	0.01	7
sh2-10	839	129697	547	100-0-0	0.01	332
sh3-10	2466	1508850	2044	100-0-0	0.80	8400

Benchmarks with Hidden Optimal Solutions

- CSP problems generated according to *model RB* (Xu 2005)
- Instances in phase transition area
- Proven to be hard

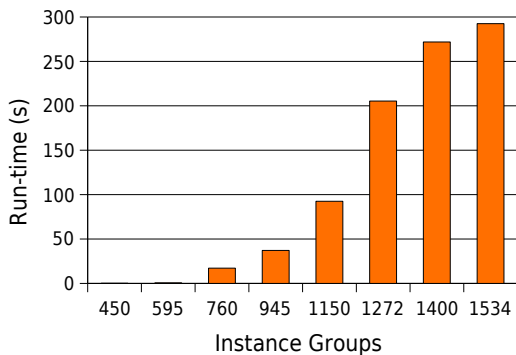
Problem instances from Xu's benchmark repository

Experimental Results IIa



- Problem sizes: $V = 450 - 1\,534$, $E = 17\,827 - 125\,982$
- All non-optimal solutions off by 1
- Found optimal solution at least once

Experimental Results IIa



- Problem sizes: $V = 450 - 1\,534$, $E = 17\,827 - 125\,982$
- All non-optimal solutions off by 1
- Found optimal solution at least once

Comparison on the BHOSLIB problems:

- Ant Colony (Gilmour & Dras, 2006): off by 8.625 on avg.
- COVER: off by 0.25

In addition: 20-year challenge problem

- $|V| = 4000$, $|E| = 572\,774$, $k^* = 3900$
- best known solution so far: 3904 vertices in 3 743 sec.
- COVER: 3903 vertices in 71 sec.

Second DIMACS Implementation Challenge (1992-1993)

80 problems from a variety of applications
up to 4000 vertices, 4 000 000 edges

- *C-fat* family: fault diagnosis
- *johnson, hamming*: coding theory
- random graphs
- *brock* family: “hard” graphs
- ...

Comparison against DLS-MC (Pullan & Hoos 2006)

Experimental Results IIIa

- For 74 of the 80 instances, COVER finds an optimal or best-known solution.
- The 6 problematic instances are all from the same problem class (brock-family).
- On 2 instances, COVER sets new records.

Experimental Results IIb

Instance	Graph		COVER			DLS-MC	
	$ V $	k^*	Quality	Median	Avg.	Quality	Avg.
c-fat200-1	200	188	100-0-0	0.01	0.01	100-0-0	0.01
c-fat200-2	200	176	100-0-0	0.01	0.01	100-0-0	0.01
keller4	171	160	100-0-0	0.01	0.01	100-0-0	0.01
keller5	776	749	100-0-0	0.03	0.07	100-0-0	0.02
keller6	3361	3302	100-0-0	15.18	15.63	100-0-0	170.48
C1000.9	1000	932	100-0-0	3.27	5.82	100-0-0	4.44
C2000.5	2000	1984	100-0-0	1.84	3.78	100-0-0	0.97
C2000.9	2000	1922	84-16-0	323.11	369.33	93-7-0	193.22
C4000.5	4000	3982	98-2-0	621.38	689.74	100-0-0	181.23
MANN_a9	45	29	100-0-0	0.01	0.01	100-0-0	0.01
MANN_a27	378	252	100-0-0	0.01	0.01	100-0-0	0.05
MANN_a45	1035	690	41-59-0	(0.28)	n/a	0-100-0	n/a
MANN_a81	3321	2221	4-3-93	(30.89)	n/a	0-0-100	n/a
brock200_1	200	179	100-0-0	0.01	0.01	100-0-0	0.02
brock200_2	200	188	100-0-0	0.23	0.43	100-0-0	0.02
brock400_1	400	373	0-0-100	(0.06)	n/a	100-0-0	n/a
brock400_3	400	369	60-30-10	247.87	135.26	100-0-0	0.18
brock800_1	800	777	0-0-100	(1.06)	n/a	100-0-0	n/a
brock800_2	800	776	0-0-100	(0.98)	n/a	100-0-0	n/a

Experimental Results IIb

Instance	Graph		COVER			DLS-MC	
	V	k^*	Quality	Median	Avg.	Quality	Avg.
c-fat200-1	200	188	100-0-0	0.01	0.01	100-0-0	0.01
c-fat200-2	200	176	100-0-0	0.01	0.01	100-0-0	0.01
keller4	171	160	100-0-0	0.01	0.01	100-0-0	0.01
keller5	776	749	100-0-0	0.03	0.07	100-0-0	0.02
keller6	3361	3302	100-0-0	15.18	15.63	100-0-0	170.48
C1000.9	1000	932	100-0-0	3.27	5.82	100-0-0	4.44
C2000.5	2000	1984	100-0-0	1.84	3.78	100-0-0	0.97
C2000.9	2000	1922	84-16-0	323.11	369.33	93-7-0	193.22
C4000.5	4000	3982	98-2-0	621.38	689.74	100-0-0	181.23
MANN_a9	45	29	100-0-0	0.01	0.01	100-0-0	0.01
MANN_a27	378	252	100-0-0	0.01	0.01	100-0-0	0.05
MANN_a45	1035	690	41-59-0	(0.28)	n/a	0-100-0	n/a
MANN_a81	3321	2221	4-3-93	(30.89)	n/a	0-0-100	n/a
brock200_1	200	179	100-0-0	0.01	0.01	100-0-0	0.02
brock200_2	200	188	100-0-0	0.23	0.43	100-0-0	0.02
brock400_1	400	373	0-0-100	(0.06)	n/a	100-0-0	n/a
brock400_3	400	369	60-30-10	247.87	135.26	100-0-0	0.18
brock800_1	800	777	0-0-100	(1.06)	n/a	100-0-0	n/a
brock800_2	800	776	0-0-100	(0.98)	n/a	100-0-0	n/a

Experimental Results IIb

Instance	Graph		COVER			DLS-MC	
	V	k^*	Quality	Median	Avg.	Quality	Avg.
c-fat200-1	200	188	100-0-0	0.01	0.01	100-0-0	0.01
c-fat200-2	200	176	100-0-0	0.01	0.01	100-0-0	0.01
keller4	171	160	100-0-0	0.01	0.01	100-0-0	0.01
keller5	776	749	100-0-0	0.03	0.07	100-0-0	0.02
keller6	3361	3302	100-0-0	15.18	15.63	100-0-0	170.48
C1000.9	1000	932	100-0-0	3.27	5.82	100-0-0	4.44
C2000.5	2000	1984	100-0-0	1.84	3.78	100-0-0	0.97
C2000.9	2000	1922	84-16-0	323.11	369.33	93-7-0	193.22
C4000.5	4000	3982	98-2-0	621.38	689.74	100-0-0	181.23
MANN_a9	45	29	100-0-0	0.01	0.01	100-0-0	0.01
MANN_a27	378	252	100-0-0	0.01	0.01	100-0-0	0.05
MANN_a45	1035	690	41-59-0	(0.28)	n/a	0-100-0	n/a
MANN_a81	3321	2221	4-3-93	(30.89)	n/a	0-0-100	n/a
brock200_1	200	179	100-0-0	0.01	0.01	100-0-0	0.02
brock200_2	200	188	100-0-0	0.23	0.43	100-0-0	0.02
brock400_1	400	373	0-0-100	(0.06)	n/a	100-0-0	n/a
brock400_3	400	369	60-30-10	247.87	135.26	100-0-0	0.18
brock800_1	800	777	0-0-100	(1.06)	n/a	100-0-0	n/a
brock800_2	800	776	0-0-100	(0.98)	n/a	100-0-0	n/a

Experimental Results IIb

Instance	Graph		COVER			DLS-MC	
	V	k^*	Quality	Median	Avg.	Quality	Avg.
c-fat200-1	200	188	100-0-0	0.01	0.01	100-0-0	0.01
c-fat200-2	200	176	100-0-0	0.01	0.01	100-0-0	0.01
keller4	171	160	100-0-0	0.01	0.01	100-0-0	0.01
keller5	776	749	100-0-0	0.03	0.07	100-0-0	0.02
keller6	3361	3302	100-0-0	15.18	15.63	100-0-0	170.48
C1000.9	1000	932	100-0-0	3.27	5.82	100-0-0	4.44
C2000.5	2000	1984	100-0-0	1.84	3.78	100-0-0	0.97
C2000.9	2000	1922	84-16-0	323.11	369.33	93-7-0	193.22
C4000.5	4000	3982	98-2-0	621.38	689.74	100-0-0	181.23
MANN_a9	45	29	100-0-0	0.01	0.01	100-0-0	0.01
MANN_a27	378	252	100-0-0	0.01	0.01	100-0-0	0.05
MANN_a45	1035	690	41-59-0	(0.28)	n/a	0-100-0	n/a
MANN_a81	3321	2221	4-3-93	(30.89)	n/a	0-0-100	n/a
brock200_1	200	179	100-0-0	0.01	0.01	100-0-0	0.02
brock200_2	200	188	100-0-0	0.23	0.43	100-0-0	0.02
brock400_1	400	373	0-0-100	(0.06)	n/a	100-0-0	n/a
brock400_3	400	369	60-30-10	247.87	135.26	100-0-0	0.18
brock800_1	800	777	0-0-100	(1.06)	n/a	100-0-0	n/a
brock800_2	800	776	0-0-100	(0.98)	n/a	100-0-0	n/a

Experimental Results IIb

Instance	Graph		COVER			DLS-MC	
	V	k^*	Quality	Median	Avg.	Quality	Avg.
c-fat200-1	200	188	100-0-0	0.01	0.01	100-0-0	0.01
c-fat200-2	200	176	100-0-0	0.01	0.01	100-0-0	0.01
keller4	171	160	100-0-0	0.01	0.01	100-0-0	0.01
keller5	776	749	100-0-0	0.03	0.07	100-0-0	0.02
keller6	3361	3302	100-0-0	15.18	15.63	100-0-0	170.48
C1000.9	1000	932	100-0-0	3.27	5.82	100-0-0	4.44
C2000.5	2000	1984	100-0-0	1.84	3.78	100-0-0	0.97
C2000.9	2000	1922	84-16-0	323.11	369.33	93-7-0	193.22
C4000.5	4000	3982	98-2-0	621.38	689.74	100-0-0	181.23
MANN_a9	45	29	100-0-0	0.01	0.01	100-0-0	0.01
MANN_a27	378	252	100-0-0	0.01	0.01	100-0-0	0.05
MANN_a45	1035	690	41-59-0	(0.28)	n/a	0-100-0	n/a
MANN_a81	3321	2221	4-3-93	(30.89)	n/a	0-0-100	n/a
brock200_1	200	179	100-0-0	0.01	0.01	100-0-0	0.02
brock200_2	200	188	100-0-0	0.23	0.43	100-0-0	0.02
brock400_1	400	373	0-0-100	(0.06)	n/a	100-0-0	n/a
brock400_3	400	369	60-30-10	247.87	135.26	100-0-0	0.18
brock800_1	800	777	0-0-100	(1.06)	n/a	100-0-0	n/a
brock800_2	800	776	0-0-100	(0.98)	n/a	100-0-0	n/a

COVER's empirical performance

- Biology: significantly faster than P-FPT
- BHOSLIB: very good performance
 - New record on 20-year challenge problem
- DIMACS: competitive with state of the art
 - New records on 2 problems
 - Bad results on brock instances

COVER is an algorithm for k -vertex cover. . .

What if optimal value for k is not known?

A small experiment:

- Start with a greedy approximation to k^*
- Search iteratively with decreasing values of k
- Conjecture: most time is spent in *last* iteration ($k = k^*$)

Search without parameter

Graph	Run-time(s)	k^*	$k^* + 1$	$k^* + 2$	$k^* + 3$	$> k^* + 3$
Globin7	0.54	48.15%	48.15%	3.70%		
Sh2-5	0.70	37.14%	37.14%	25.71%		
johnson32-2-4	0.92	28.26%	28.26%	26.09%	11.96%	5.43%
brock200_1	1.52	21.71%	17.11%	17.11%	16.45%	27.63%
p_hat700-1	2.16	43.06%	24.54%	11.57%	10.65%	10.19%
keller5	5.12	40.43%	7.23%	5.08%	5.08%	42.19%
frb30-15-1	7.15	41.68%	24.20%	7.27%	3.64%	23.22%
hamming10-4	10.73	47.62%	25.63%	2.42%	2.42%	21.90%
san400_0.5_1	15.30	34.77%	22.35%	15.82%	19.28%	7.78%
DSJC1000.5	88.10	97.63%	1.16%	0.30%	0.30%	0.62%
brock200_3	166.01	99.18%	0.19%	0.16%	0.16%	0.32%
san1000	645.26	28.25%	20.13%	18.73%	18.72%	14.16%
frb50-23-4	1344.88	85.85%	9.78%	2.42%	1.42%	0.53%
frb59-26-5	17611.90	84.57%	13.84%	0.89%	0.35%	0.35%

Conclusion:

A very competitive vertex cover algorithm

- Simple, needs no parameter
- Performs well even if k^* is not known

Future work:

Further techniques to escape local minima

Thank you!
Questions?