Lock-chart solving

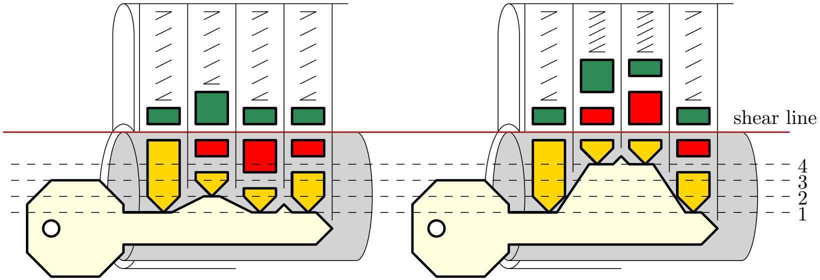
Dissertation defense

Radomír Černoch

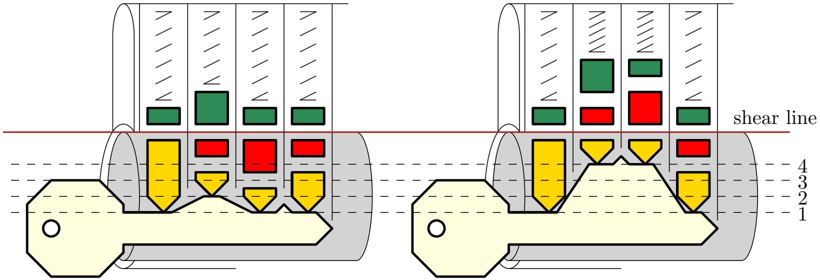
27/6/2018

1. Introduction

Tumbler lock



Tumbler lock

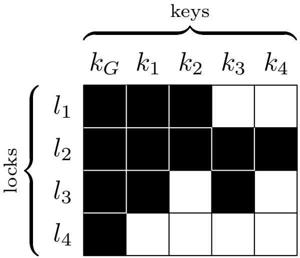


In this picture:

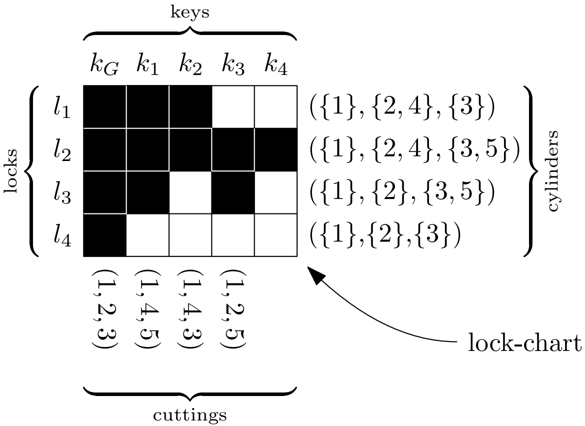
- There are 4 *positions* (vertical bars).
- There are 4 *cutting depths* (horizontal lines).

$$p = 4, d = 4$$

Lock-chart problem



Lock-chart problem



Motivation

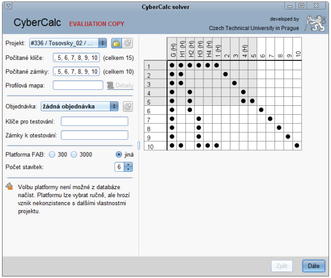
Reviewer's comment

Based on the presented research, has there been a progamme written that could be practically used to design master keyed systems?

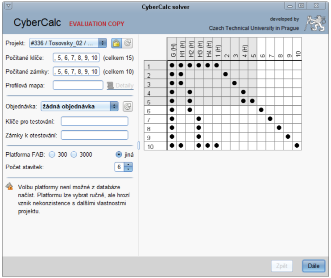
Otázka oponenta

Byl na základě představeného výzkumu napsán program, který se v praxi použil na návrh zámků a klíčů, případně na nějaké dílčí úlohy?

Motivation

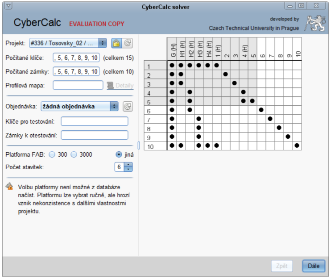

CyberCalc solver
CyberCalc EVALUATION COPY
developed by
Czech Technical University in Prague
Projekt: #336 / Tosovsky_02 / ...
Počítané klíče: , 5, 6, 7, 8, 9, 10 (celkem 15)
Počítané zámky: , 5, 6, 7, 8, 9, 10 (celkem 10)
Profilová mapa:
Detaily
Objednávka: žádná objednávka
Klíče pro testování:
Zámky k otestování:
Platforma FAB: 300 3000 jiná
Počet stavítek: 6
Volbu platformy není možné z databáze načíst. Platformu lze vybrat ručně, ale hrozí vznik nekonzistence s dalšími vlastnostmi projektu.
Zpět
Dále

Motivation



CyberCalc: Depth-first-search with constraint-satisfaction pruning and many heuristics.

Motivation

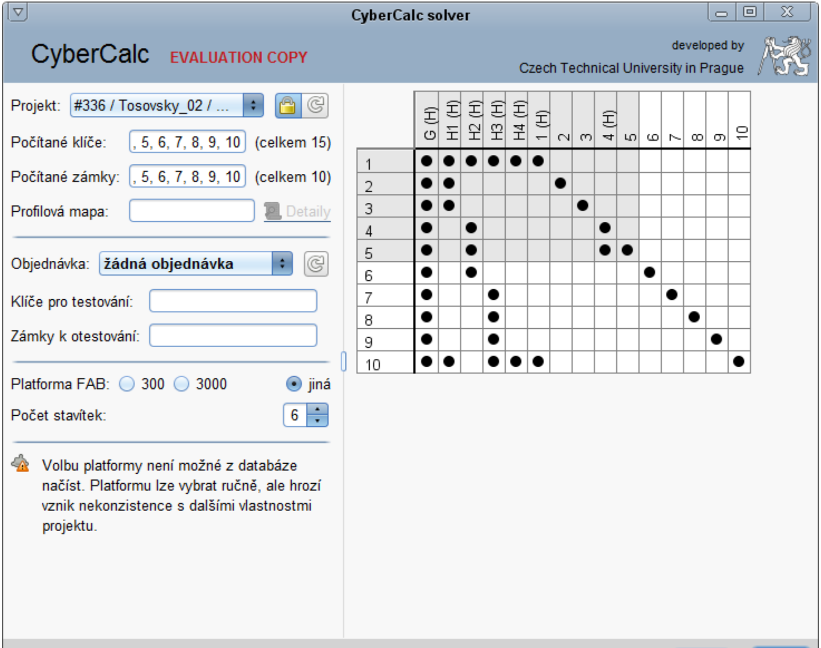


CyberCalc: Depth-first-search with constraint-satisfaction pruning and many heuristics.

Issues

- **Lack of insight** into heuristics' effects.
- Does not scale for **modern platforms** with $\geq 10^6$ key cuttings.
- Can some subproblems be **polynomial**? Under which assumptions?
- The **extensibility** problem.

Motivation


CyberCalc solver
CyberCalc
EVALUATION COPY
developed by
Czech Technical University in Prague
Projekt:
#336 / Tosovsky_02 / ...
Počítané klíče:
, 5, 6, 7, 8, 9, 10
(celkem 15)
Počítané zámky:
, 5, 6, 7, 8, 9, 10
(celkem 10)
Profilová mapa:
Detaily
Objednávka:
žádná objednávka
Klíče pro testování:
Zámky k otestování:
Platforma FAB:
300
3000
jiná
Počet stavítek:
6
Volbu platformy není možné z databáze načíst. Platformu lze vybrat ručně, ale hrozí vznik nekonzistence s dalšími vlastnostmi projektu.
G (H)
H1 (H)
H2 (H)
H3 (H)
H4 (H)
1 (H)
2
3
4 (H)
5
6
7
8
9
10
1
2
3
4
5
6
7
8
9
10

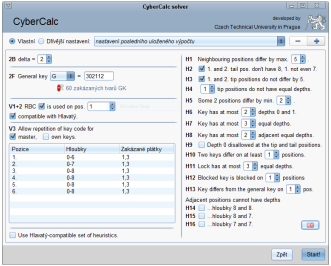
2. Complexity classes

Frameworks

4 constraint *frameworks* of increasing expressivity:

1. **Vanilla**: Number of positions p and cutting depths d.
2. (**Asymmetric**: Number of cutting depths varies between positions.)
3. **General**: Plus a set of *general constraints* (gecons).
4. (**Explicit**: List of valid key cuttings is algorithm's input.)

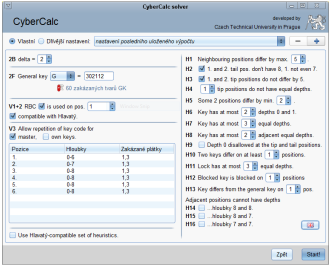
General framework



Why gecons?

- Most constraints "overlap", **counting key cuttings** that satisfy all constraints is difficult.
- **Gecon** = a formally defined group of forbidden key cuttings.
- 15 rules can be each "compiled" to a set of gecons.

General framework



Counting key cuttings

- Derive number of key cuttings **invalidated** by 1 gecon.
- Define an **intersection** of 2 gecons (which is also a gecon).
- Design a **inclusion-exclusion** counting procedure.
- It can count $\approx 840 \cdot 10^6$ keys within $60\,$s.

General framework

$\mathcal{NP}$-completeness

- **Gecon** corresponds to 1 clause in the SAT problem.
- SAT is equivalent to solving a "1 key $\times$ 0 locks" lock-chart.
- Lock-chart solving in the general framework is $\mathcal{NP}$-complete.

General framework

$\mathcal{NP}$-completeness

- **Gecon** corresponds to 1 clause in the SAT problem.
- SAT is equivalent to solving a "1 key $\times$ 0 locks" lock-chart.
- Lock-chart solving in the general framework is $\mathcal{NP}$-complete.

Counting key cuttings

- Both reductions **preserve** the number of solutions.
- $\#$SAT counts the number of valid key cuttings.
- Counting key cuttings is in $\#\mathcal{P}$.

$\mathcal{NP}$-complete classes

- = lock-chart problem instance grows in this parameter

	framework	lock-chart	positions	depths	gecons	keys	locks
1.	general	all	•		•		

$\mathcal{NP}$-complete classes

• = lock-chart problem instance grows in this parameter

	framework	lock-chart	positions	depths	gecons	keys	locks	
1.	general	all	•		•			
2.	vanilla	extension	•				•	Lawer (2004)

$\mathcal{NP}$-complete classes

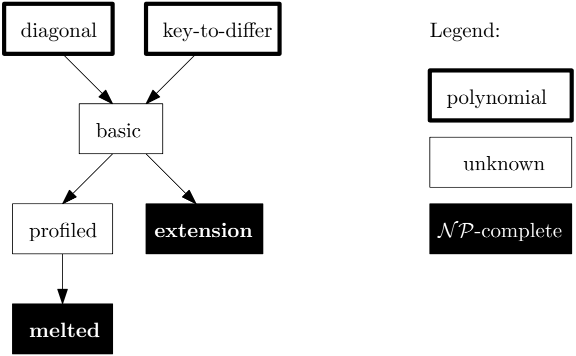
• = lock-chart problem instance grows in this parameter

	framework	lock-chart	positions	depths	gecons	keys	locks	
1.	general	all	•		•			
2.	vanilla	extension	•				•	Lawer (2004)
3.	vanilla	melted-profiles		•		•	•	

- Results 1. and 2. proved by a reduction from SAT.
- Result 3. by a reduction from k-coloring.

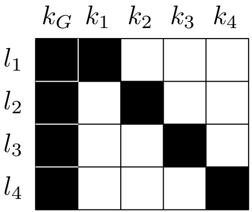
Vanilla framework

Complexity classes of lock-chart problem variants:



Vanilla framework

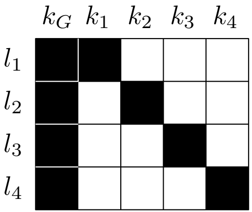
Diagonal lock-chart:



Contains exactly 1 *general* key which opens all locks and several *individual* keys which open exactly 1 lock.

Vanilla framework

Diagonal lock-chart:



Contains exactly 1 *general* key which opens all locks and several *individual* keys which open exactly 1 lock.

Properties

- Finding cuttings for individual keys translates to the *maximum independent set* (MIS) problem.
- This is true for all frameworks.

Vanilla framework

Structure of the MIS graph

- The MIS graph can be partitioned into $S_0, \ldots, S_p$ **classes**, based on the distance to the cutting of the general key.
- Size of the q-th class was derived to be

$$|S_q| = \binom{p}{p-q} \cdot (d-1)^{p-q} \ ,$$

maximized if $q = \hat{q}$, where

$$\hat{q} = \left\lfloor \frac{p+1}{d} \right\rfloor \ .$$

Vanilla framework

Structure of the MIS graph

- The MIS graph can be partitioned into $S_0, \ldots, S_p$ **classes**, based on the distance to the cutting of the general key.
- Size of the q-th class was derived to be

$$|S_q| = \binom{p}{p-q} \cdot (d-1)^{p-q} \ ,$$

  maximized if $q = \hat{q}$, where

$$\hat{q} = \left\lfloor \frac{p+1}{d} \right\rfloor \ .$$

- Each S_q class is an independent set (true in every fw.).
- $S_{\hat{q}}$ is the maximum independent set (in vanilla fw.).

Vanilla framework

Structure of the MIS graph

- The MIS graph can be partitioned into $S_0, \ldots, S_p$ **classes**, based on the distance to the cutting of the general key.
- Size of the q-th class was derived to be

$$|S_q| = \binom{p}{p-q} \cdot (d-1)^{p-q} \ ,$$

maximized if $q = \hat{q}$, where

$$\hat{q} = \left\lfloor \frac{p+1}{d} \right\rfloor \ .$$

- Each S_q class is an independent set (true in every fw.).
- $S_{\hat{q}}$ is the maximum independent set (in vanilla fw.).
- The *rotating constant method* (RCM) was analysed.

Vanilla framework

- Solution exists iff the number of keys is at most $|S_{\hat{q}}|$.
- Diagonal lock-charts in the vanilla framework are polynomial.

Other frameworks

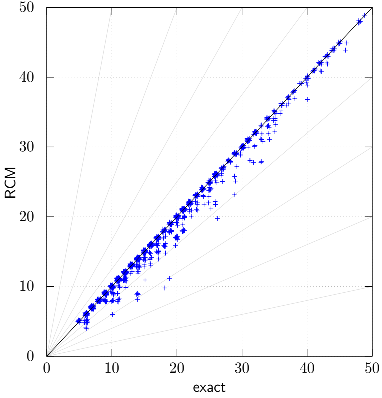
- MIS is $\mathcal{NP}$-complete, but it can be approximated.

Other frameworks

- MIS is $\mathcal{NP}$-complete, but it can be approximated.
- Since each S_q is an independent set, RCM gives at least a **lower bound** on the MIS size.
- How tight is this bound?

Other frameworks

- Real-world constraints were randomly sampled.
- The MIS graph was constructed in memory.
- RCM was compared againts an exact algorithm (backtracking, exponential runtime).



Other frameworks

RCM gives a lower bound on the size of diagonal lock-charts:

1. **Vanilla**: Given by a closed formula.
2. **Asymmetric**: Calculated using dynamic programming (not discussed here, still in $\mathcal{P}$).
3. **General**: A modified inclusion-exclusion algorithm.

3. Practical algorithms

Use of cutting-counting

A typical real-world lc.:

```
**..*.....*........
*.*..*.....*....... <- Group 1
*...**.*....*......
***.....*....*.....
****...*......*....
**..*..........*...
*.*..*..........*.. <- Group 1
*...***.*........*.
**********........*
\____ ___/\___ ___/
     ˘         ˘
  MASTER  INDIVIDUAL
```

Use of cutting-counting

A typical real-world lc.:

```
**..*.....*........
*.*..*.....*....... <- Group 1
*...**.*....*......
***.....*....*.....
****...*......*....
**..*..........*...
*.*..*..........*.. <- Group 1
*...***.*........*.
**********........*
\____ ___/\___ ___/
     ˇ         ˇ
  MASTER  INDIVIDUAL
```

- Each *group* defined by a combination of master keys.
- RCM was generalized to count individual keys in each group.

All-different pruning

- *All-different* algorithm is inspired by *constraint satifaction problems*.
- If there are less cuttings for individual keys than the size of the group, **no solution can exist**.

All-different pruning

- The algorithm by Lawer (2004) (which optimizes a criterial function) was taken as a base-line.
- The all-different pruning reduces runtime
  - by 9% on a real-world dataset,
  - by 80% on a synthetic dataset (larger lock-charts).
- Keeps completeness and optimality.

SAT approach

- Lock-chart problem was translated to SAT.
- The criterial function was approximated (no optimality guaranteed).
- The SAT instances were solved by MiniSAT Eén and Sörensson (2003).

SAT approach

- Lock-chart problem was translated to SAT.
- The criterial function was approximated (no optimality guaranteed).
- The SAT instances were solved by MiniSAT Eén and Sörensson (2003).
- Only algorithm that solves **all lock-charts** in all datasets.
- Runtime is **2 orders of magnitude faster**.
- Optimal result achieved in **all but one** lock-chart.
- Suitable for small lock-charts (up to $\sim 100 \times 100$) due to many auxiliary variables.

4. Extensibility

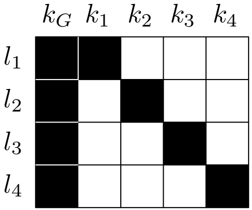
General approach

- **Extensibility:** dealing with unknown constraints, added after a solution is found.
- **Example:** Adding 1 floor to the building.

General approach

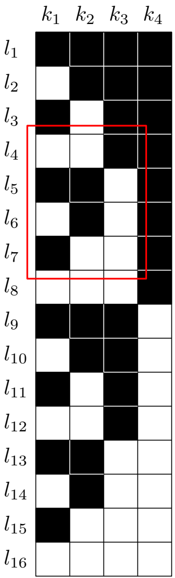
- **Extensibility:** dealing with unknown constraints, added after a solution is found.
- **Example:** Adding 1 floor to the building.
- Extensibility was vaguely defined and hard to quantify.
- Main idea: Given some assumptions, **extrapolate** the from-scratch lock-chart to the **largest solvable** lock-chart.
- Such lock-chart is called *extremal*.

Individual keys



- **Assumptions:**
  1. The lock-chart is diagonal.
  2. Only individual keys will be added in future.
- **Consequences:**
  1. Pick cuttings from $S_{\hat{q}}$.
  2. The extremal lock-chart can have at most $|S_{\hat{q}}|$ individual keys / locks.

Independent master keys



- **Assumption:** Lock-chart has at most $p \cdot (d-1)+1$ keys.
- **Consequences:**
  1. Any lock can be added in future.
  2. There are $2^{p\cdot(d-1)+1}$ such locks.

Conclusions

- The computational complexity proof show that the $\mathcal{P} \times \mathcal{NP}$ boundary disects the lock-chart problem.
- Practical cutting-counting algorithm allowed by the "compilation to gecons".
- All-different pruning improves a state-of-the-art algorithm.
- SAT-based algorithm is the new baseline.
- Extensibility has been formalized.

Thank you for your attention.

Question #1

How many cylinders are there?

- Cutting is a p-dimensional vector with discrete values $\{1, \ldots, d\}$. There may be d^p of them.
- On every position, a cylinder may or may not have a cutting depth d. There are $p \cdot d$ such binary choices. Together $2^{p \cdot d}$ cylinders.

Question #2

Citation

15 “real” constraint can be polynomially translated to gecons.

Reviewer’s comment

For how many constraints is this true? Is there a common pattern among the remaining ones?

Otázka oponenta

Pro kolik podmínek to je pravda a co měly společné podmínky, pro které to neplatilo?

Question #3

All constraints that are translated to gecons, are translated to a polynomial number of gecons.

Example

"The first n positions may not be equal."
Generates $\mathcal{O}(d \cdot n^2)$ gecons:

$$
\begin{array}{cccc}
(1,1,*,*,\ldots,*), & (2,2,*,*,\ldots,*), & \ldots, & (d,d,*,*,\ldots,*) \\
(1,*,1,*,\ldots,*), & (2,*,2,*,\ldots,*), & \ldots, & (d,*,d,*,\ldots,*) \\
(*,1,1,*,\ldots,*), & (*,2,2,*,\ldots,*), & \ldots, & (*,d,d,*,\ldots,*) \\
\vdots & \vdots & \vdots & \vdots
\end{array}
$$

Question #3

Some constraints are not translatable to gecons.

Example

"Some two cutting depths differ by at least *n*."

Solution in the solver

We use *existential constraints* (excons) Hořeňovský (2018). Used for constraints speaking about "existence" of a cutting depth.

Question #4

Reviewer's criticism

Theoretical results are straightforward, except for Theorem 52, whose proof is incomplete.

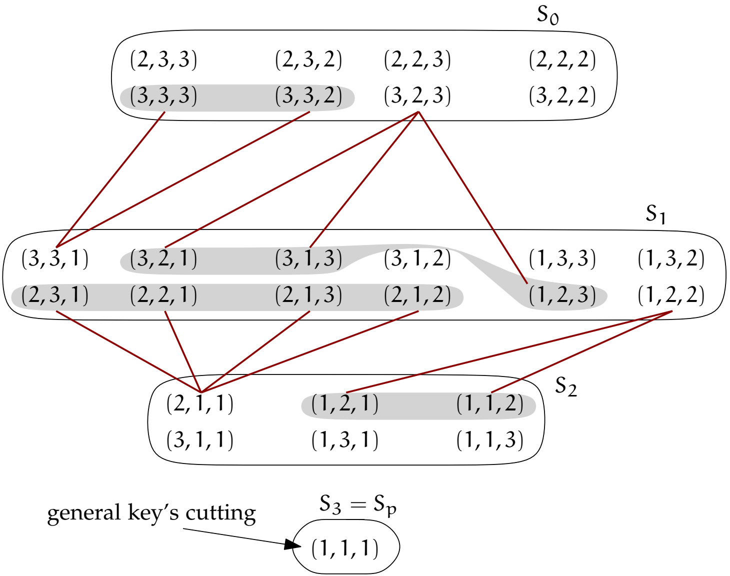
Výhrada oponenta

K teoretickým výsledkům mám výhrady: jsou povětšinou přímočaré s výjímkou věty 52, která má neúplný důkaz.

Question #4

- Proofs in the theoretical results are straightforward indeed.
- Most of them fit within a single A5 page (see page 37).
- The results are still novel.

Question #4


S_0
(2, 3, 3)
(2, 3, 2)
(2, 2, 3)
(2, 2, 2)
(3, 3, 3)
(3, 3, 2)
(3, 2, 3)
(3, 2, 2)
S_1
(3, 3, 1)
(3, 2, 1)
(3, 1, 3)
(3, 1, 2)
(1, 3, 3)
(1, 3, 2)
(2, 3, 1)
(2, 2, 1)
(2, 1, 3)
(2, 1, 2)
(1, 2, 3)
(1, 2, 2)
S_2
(2, 1, 1)
(1, 2, 1)
(1, 1, 2)
(3, 1, 1)
(1, 3, 1)
(1, 1, 3)
$S_3 = S_p$
general key's cutting
(1, 1, 1)

Question #4

- Let $0 \leq q < r \leq p$, $S'_q \subseteq S_q$, $S'_r \subseteq S_r$ and $S'_q \cup S'_r$ be an independent set.
- Proved that

$$|S'_q| + |S'_r| \leq \max\{|S_q|, |S_r|\} \ .$$

- Can $|S'_q| + |S'_r| + |S'_s| + \cdots \leq \max\{|S_q|, |S_r|, |S_s|, \ldots\}$?

Question #4

- Let $0 \leq q < r \leq p$, $S'_q \subseteq S_q$, $S'_r \subseteq S_r$ and $S'_q \cup S'_r$ be an independent set.
- Proved that

$$|S'_q| + |S'_r| \leq \max\{|S_q|, |S_r|\} \ .$$

- Can $|S'_q| + |S'_r| + |S'_s| + \cdots \leq \max\{|S_q|, |S_r|, |S_s|, \ldots\}$?
- Proof is incomplete.
- There is no known counter-example.
- I am convinced that the theorem still holds.

Question #5

Reviewer's comment

The algorithms were tested in the "vanilla framework", but the gecons alone cause NP-completeness. Are there resons to believe that the results would not change [if gecons were included]?

Otázka oponenta

Testování algoritmů proběhlo ve „vanilla framework". Samotná omezení ve formě geconů zapřičiňují NP-úplnost. Existují důvody pro to, že by se výsledky podstatně nezměnily při porovnávání vstupních dat z praxe?

Question #5

- Let's assume that the Lawers' algorithm can (somehow) accomodate gecons.
- **Will SAT be still better?**

Question #5

- Let's assume that the Lawers' algorithm can (somehow) accomodate gecons.
- **Will SAT be still better?**

1. **Runtime**:
   - Finding a valid key cutting given real-world constraints is easy (≤ 1 ms).
   - Gecons are applied to each key separately (exploited by DPLL).
   - Finding valid key cuttings for ~ 100 keys is ≤ 1 s.

Question #5

- Let's assume that the Lawers' algorithm can (somehow) accomodate gecons.
- **Will SAT be still better?**

1. **Runtime**:
   - Finding a valid key cutting given real-world constraints is easy (≤ 1 ms).
   - Gecons are applied to each key separately (exploited by DPLL).
   - Finding valid key cuttings for ~ 100 keys is ≤ 1 s.
2. **Criterial function:**
   - Some modifications to the strategy are needed.
   - It is tricky to pick a cutting depths to forbid.
   - This is done in the commercial product.
   - Lock-charts of size ~ 25 keys can still be optimized within 3 s.
3. Also, the 2 orders-of-magnitude margin is very big.

Question #0

Reviewer's comment

Based on the presented research, has there been a progamme written that could be practically used to design master keyed systems?

Otázka oponenta

Byl na základě představeného výzkumu napsán program, který se v praxi použil na návrh zámků a klíčů, případně na nějaké dílčí úlohy?

Question #0

- Yes.
- The translation to SAT is already in production.
- A fully fledged all-different pruning is in development.

Eén, N. and Sörensson, N. 2003. An extensible sat-solver. In *Theory and applications of satisfiability testing*, pages 502–518. Springer.

Hořeňovský, M. 2018. Performance analysis of a master-key system solver. Master's thesis, Czech Technical University in Prague.

Lawer, A. 2004. Calculation of lock systems. Master's thesis, Royal Institute of Technology.