KU LEUVEN

Guided Bottom-up Interactive Constraint Acquisition

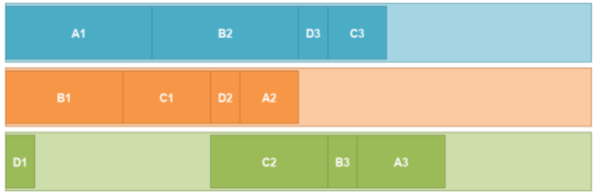
Dimos Tsouros, Senne Berden, Tias Guns
dimos.tsouros@kuleuven.be, senne.berden@kuleuven.be, tias.guns@kuleuven.be

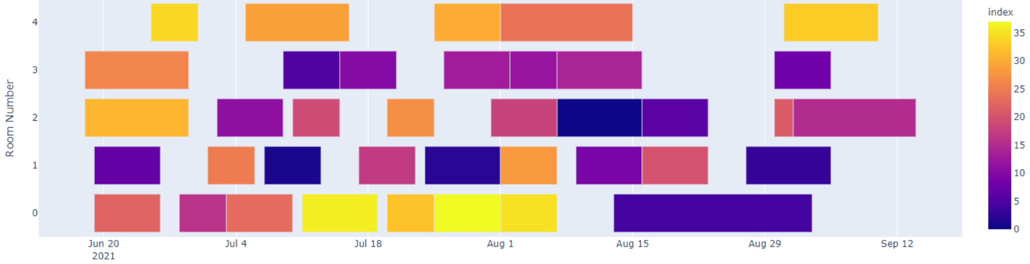
29/08/2023

KU LEUVEN

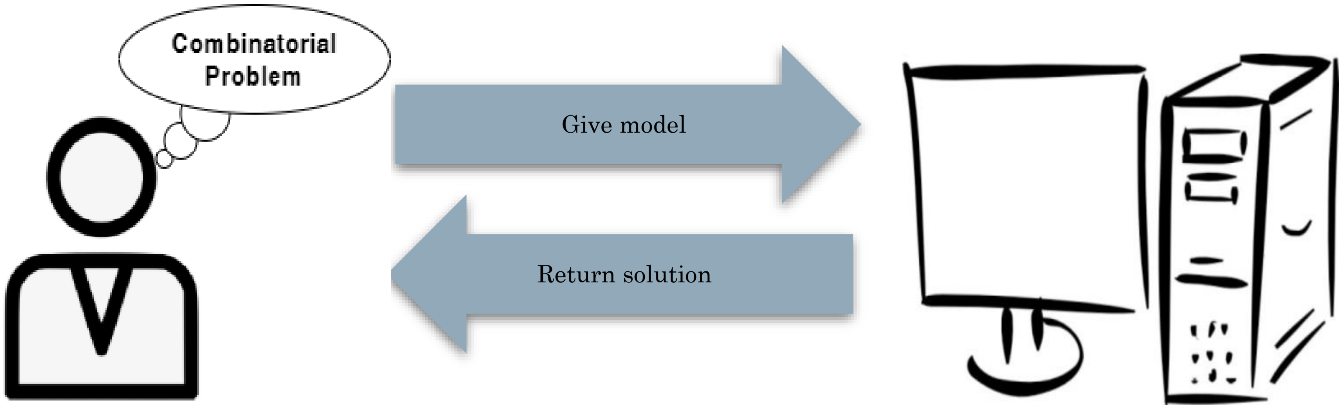
Introduction

- Constraint programming (CP)
  - Solving combinatorial problems in AI





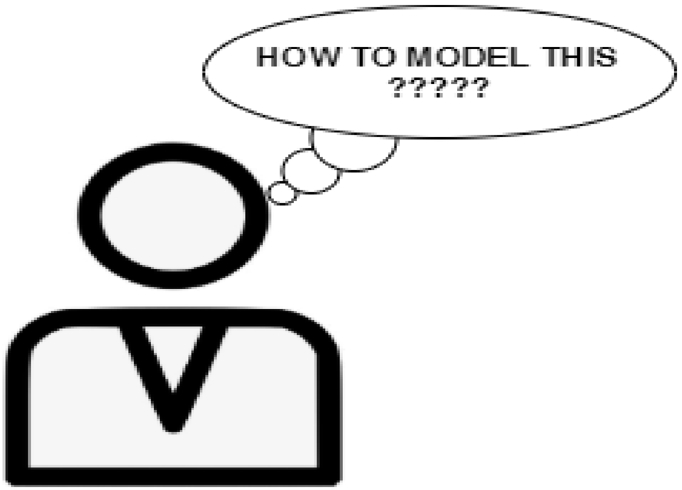
- Model + Solve paradigm



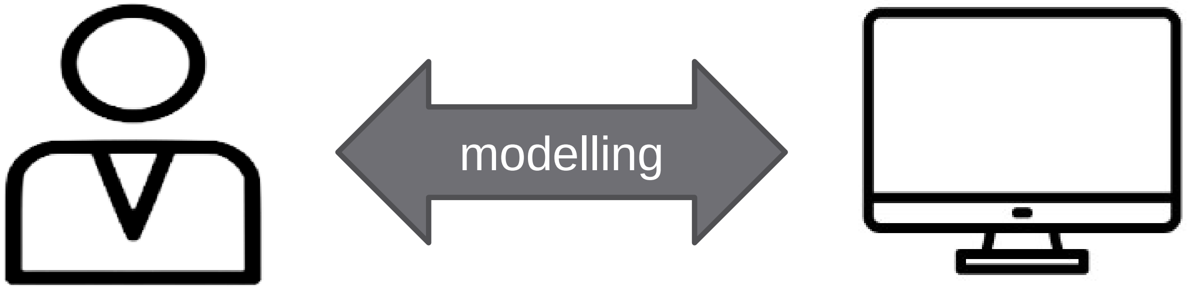
Introduction

Modelling is not always trivial

- Requires expertise
- Bottleneck for the wider use of CP

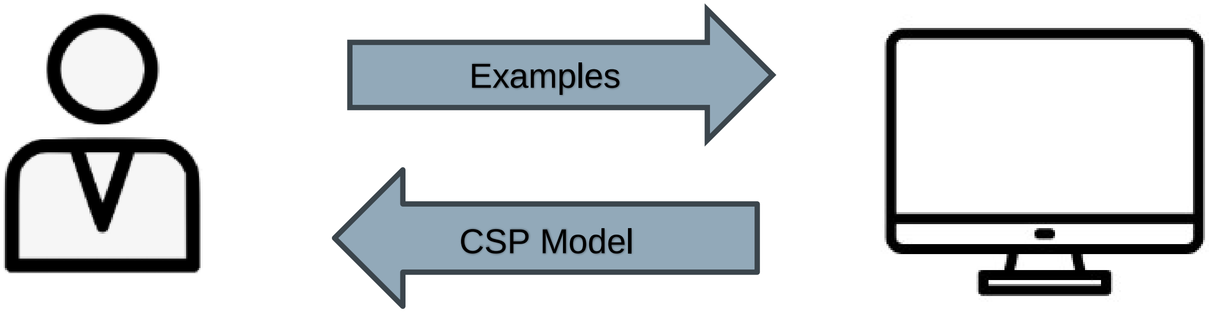


Constraint Acquisition

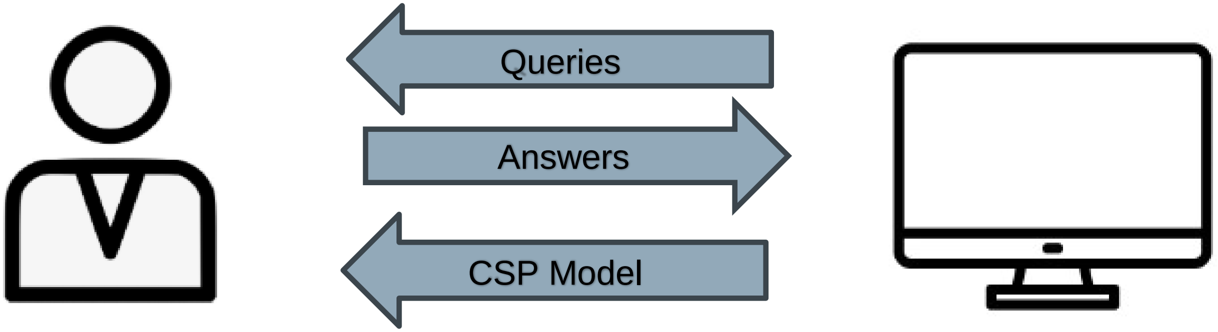


Introduction (4/4)

Passive acquisition: Using existing data



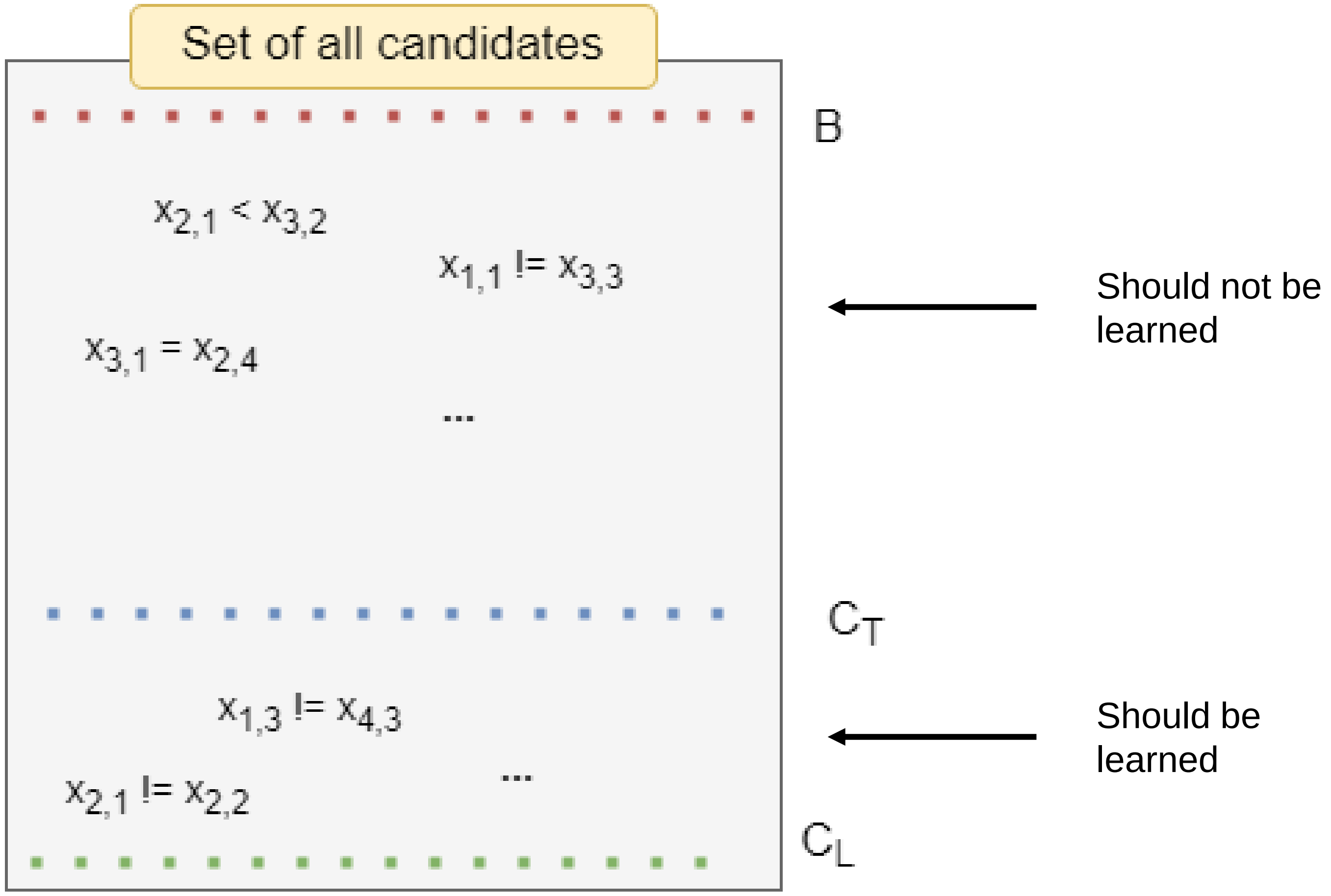
(Inter)active acquisition: Interact with the user



Constraint Acquisition, C. Bessiere et al., AIJ, 2017

Adapting Candidate Elimination

- **B**: set of (remaining) candidate constraints
- **C_T**: target set of constraints
- **C_L**: learned set of constraints

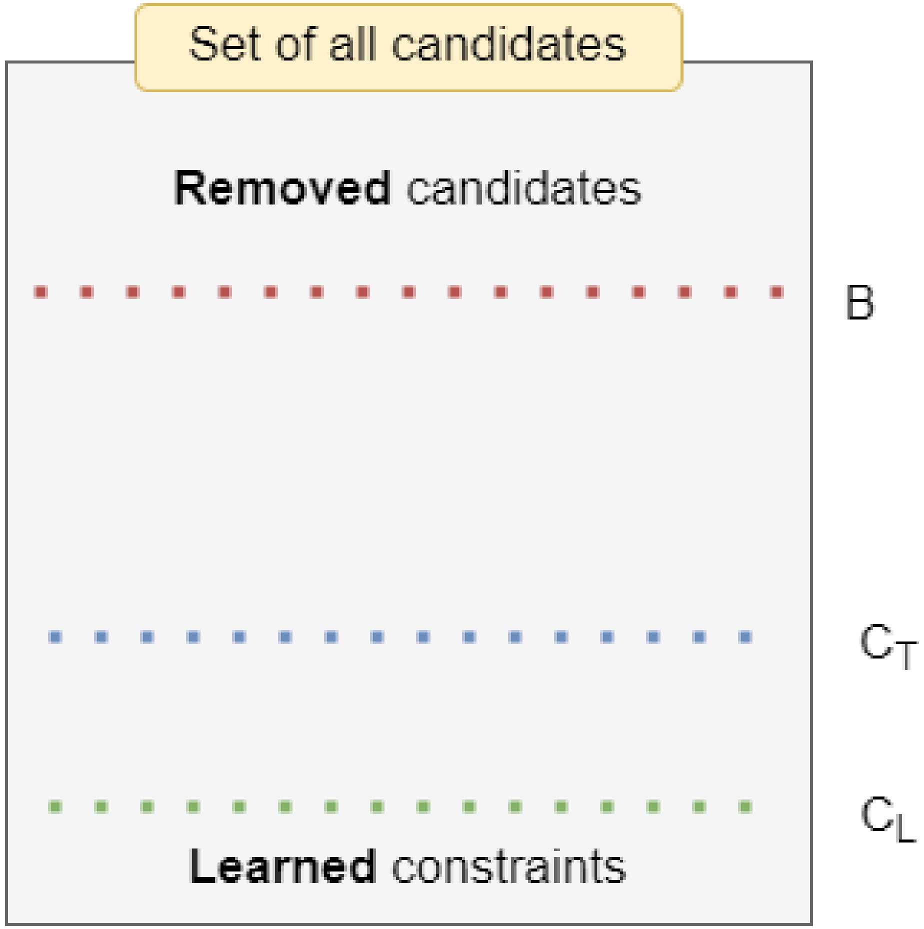


Adapting Candidate Elimination

During the learning process:

- Constraints are removed from *B*
- Constraints are added to C_L

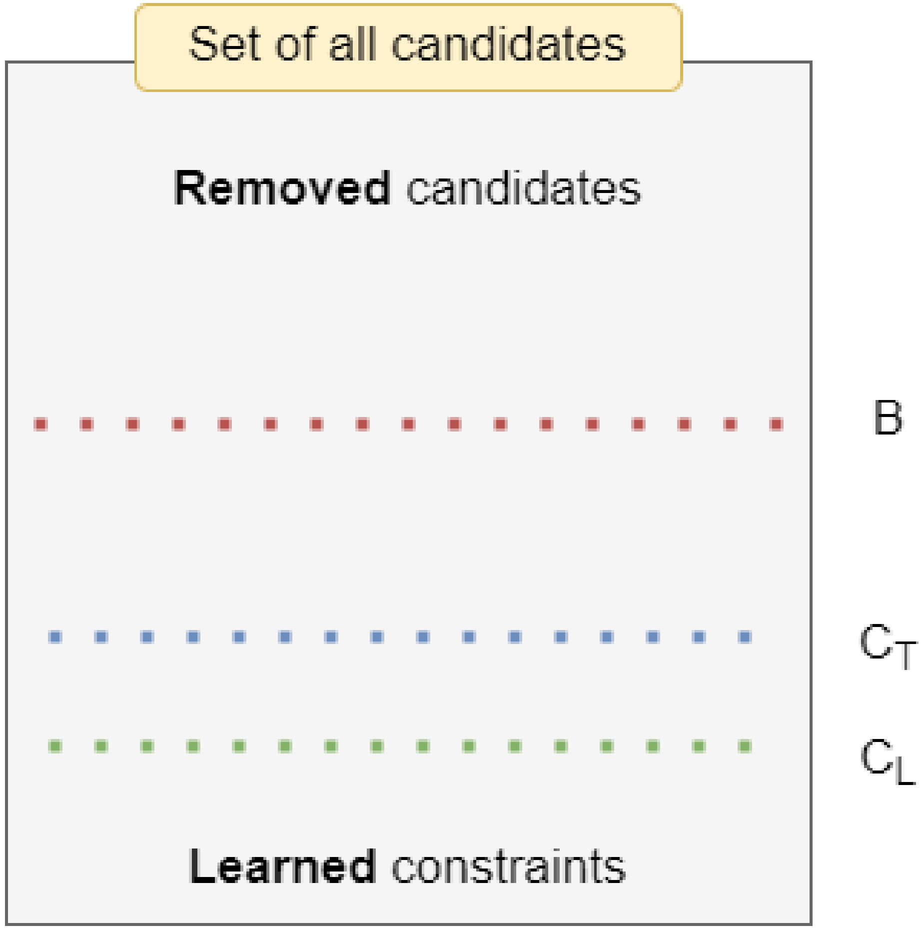
- **B**: set of (remaining) candidate constraints
- **C_T**: target set of constraints
- **C_L**: learned set of constraints



Adapting Candidate Elimination

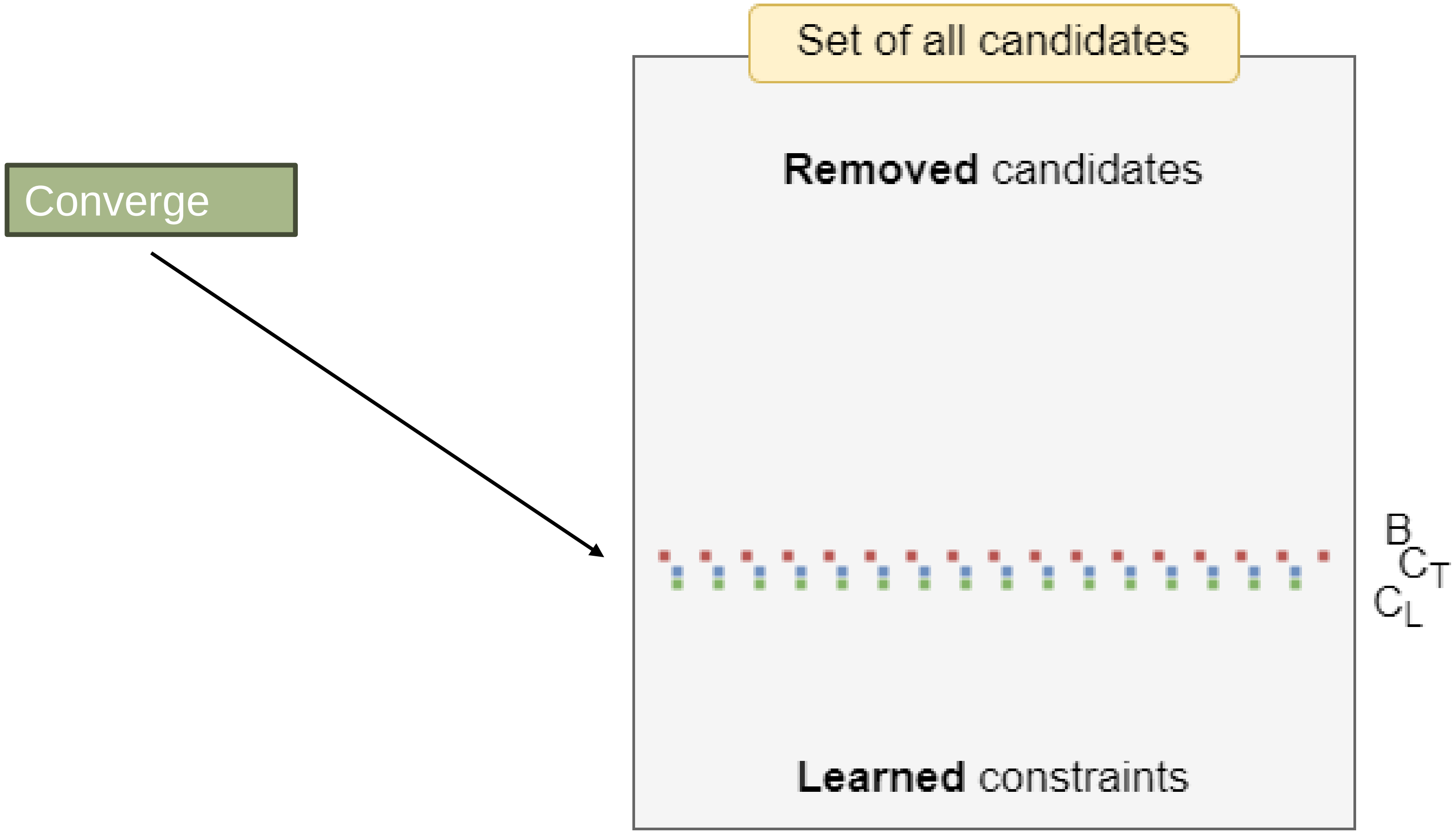
During the learning process:

- Constraints are removed from *B*
- Constraints are added to C_L



- **B**: set of (remaining) candidate constraints
- $\mathbf{C_T}$: target set of constraints
- $\mathbf{C_L}$: learned set of constraints

Adapting Candidate Elimination



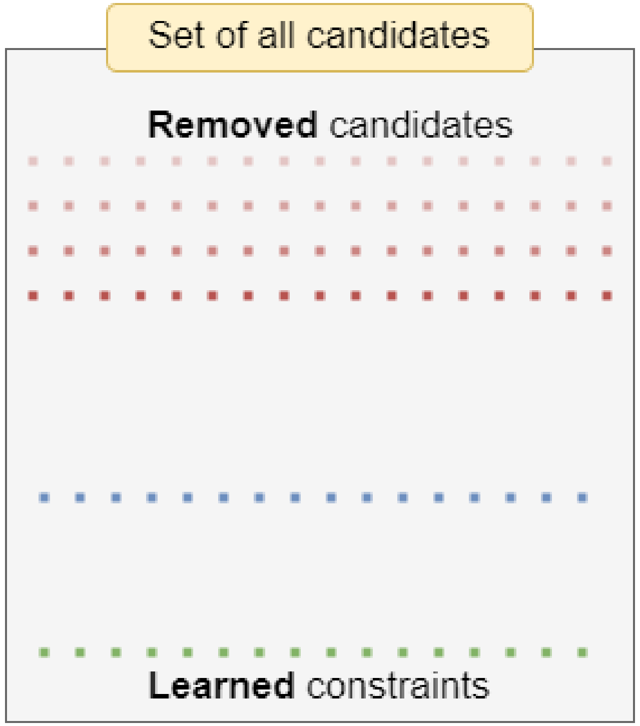
Adapting Candidate Elimination

- Examples: Assignments to the variables of the problem

- Learning from ***positive*** examples (Solutions):

- Violated constraints cannot be part of the model
- Otherwise, it could not be a solution

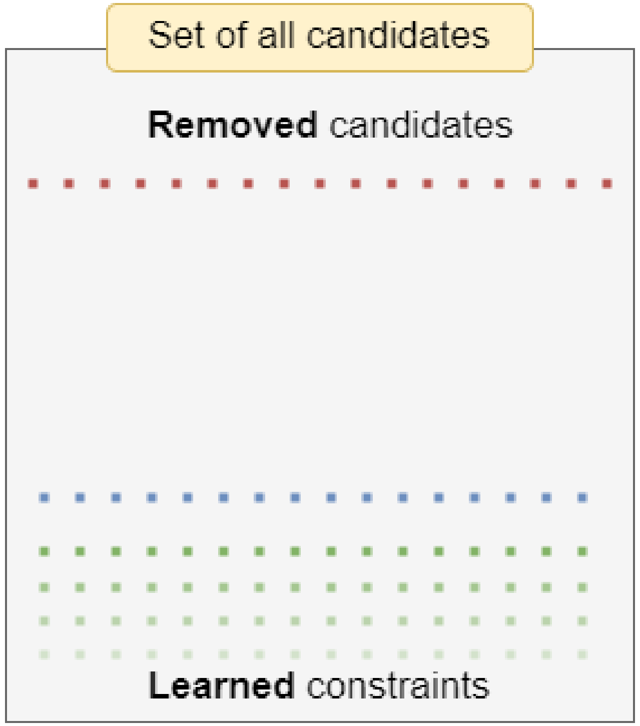
Shrinking the bias



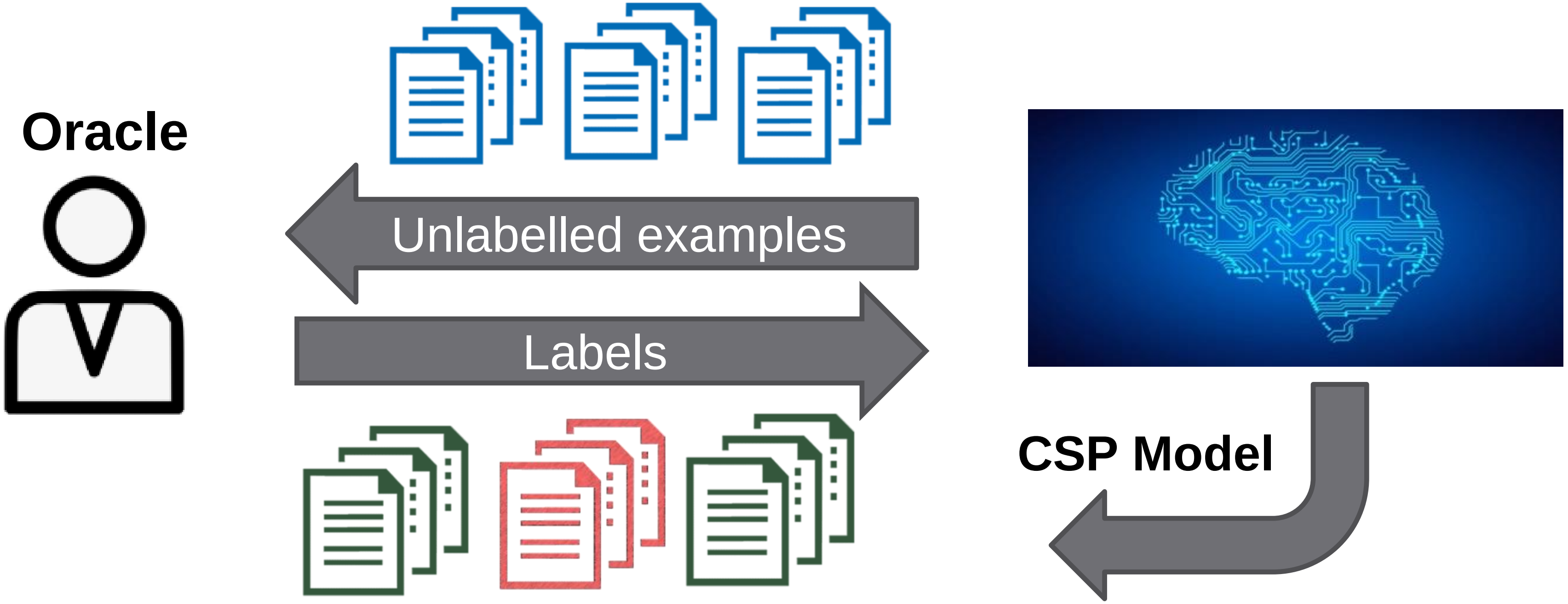
- Learning from ***negative*** examples (Non-solutions):

- One (or more) violated constraint is a constraint of the problem
- Otherwise, it would be a solution

Learning Constraints



Interactive Constraint Acquisition



Membership query

1	1	3	4
3	2	1	1
2	2	3	1
2	3	4	3

Answer: Negative in both of them (a constraint is violated)

Partial query

1	1	-	4
3	-	1	-
-	-	-	-
2	-	-	-

Learning Constraints through Partial Queries, C. Bessiere et al., AIJ, 2023

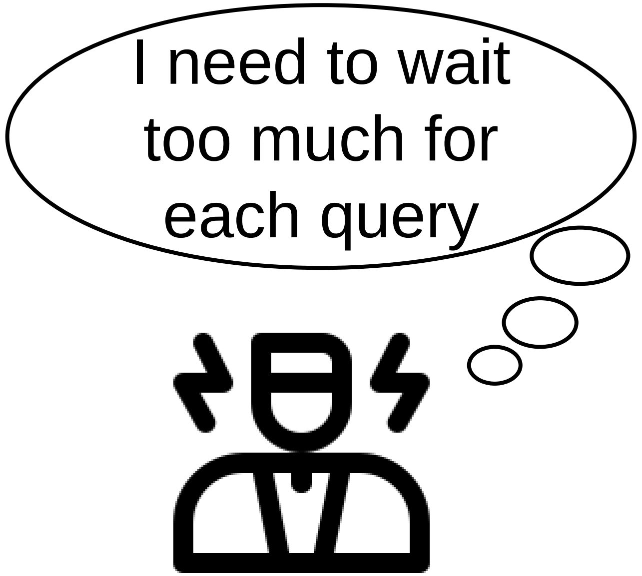
KU LEUVEN

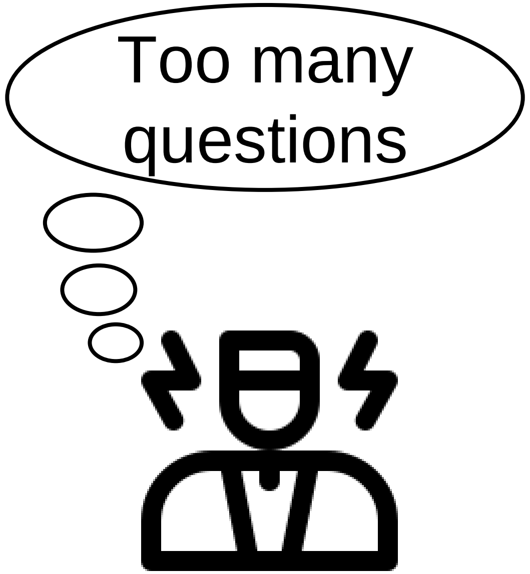
Challenges for interactive CA

Large query generation times – premature convergence – custom Solvers

Handling of large sets of candidate constraints

Number of queries





Learning constraint models from data, D Tsouros et al., 2023

Contributions

Large query generation times – premature convergence – custom Solvers

Handling of large sets of candidate constraints

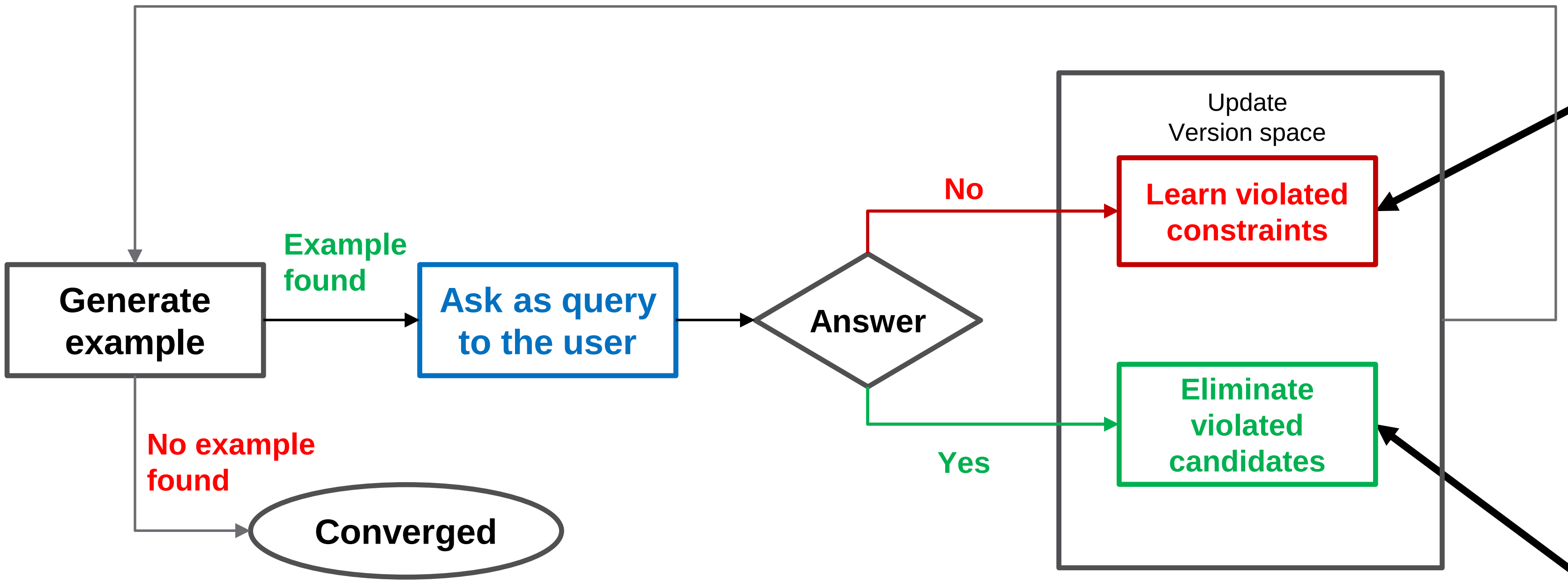
Number of queries

Projection-Based Query Generation (PQ-Gen) using conventional solvers

Consider parts of the problem in each iteration, in a bottom-up procedure (GrowAcq)

Guide query generation to query to generate better queries

Interactive Constraint Acquisition



Learn constraints: zoom in violated constraints' scopes using partial queries and add them to C_L

Shrink version space: remove violated constraints from B

Learning Constraints through Partial Queries, C. Bessiere et al., AIJ, 2023

Query generation

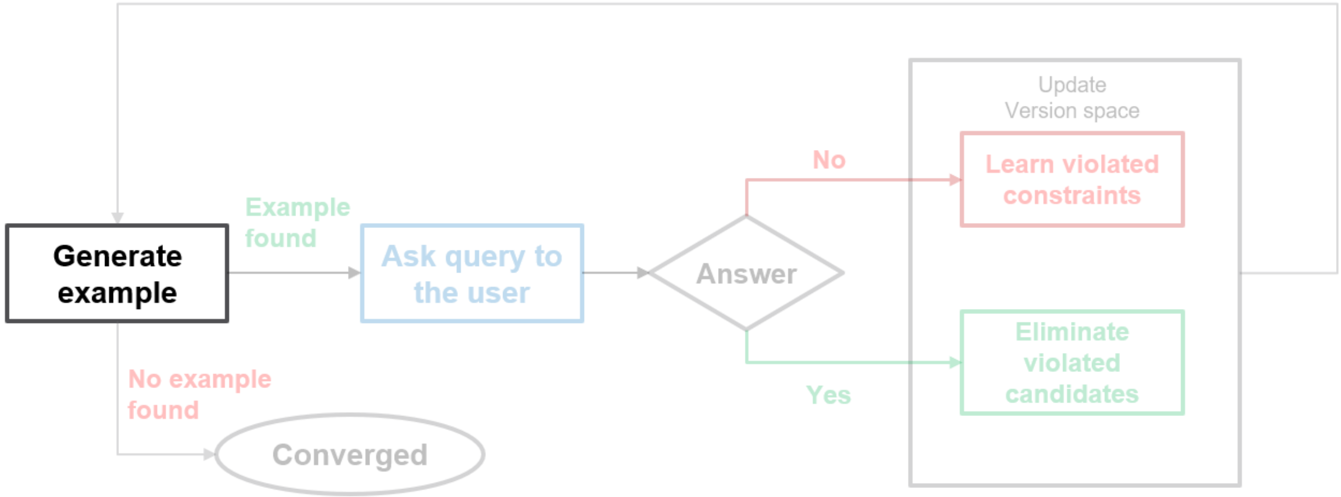
Informative query

- Generate “informative” examples

Quality of query

- Get the maximum amount of information

Convergence



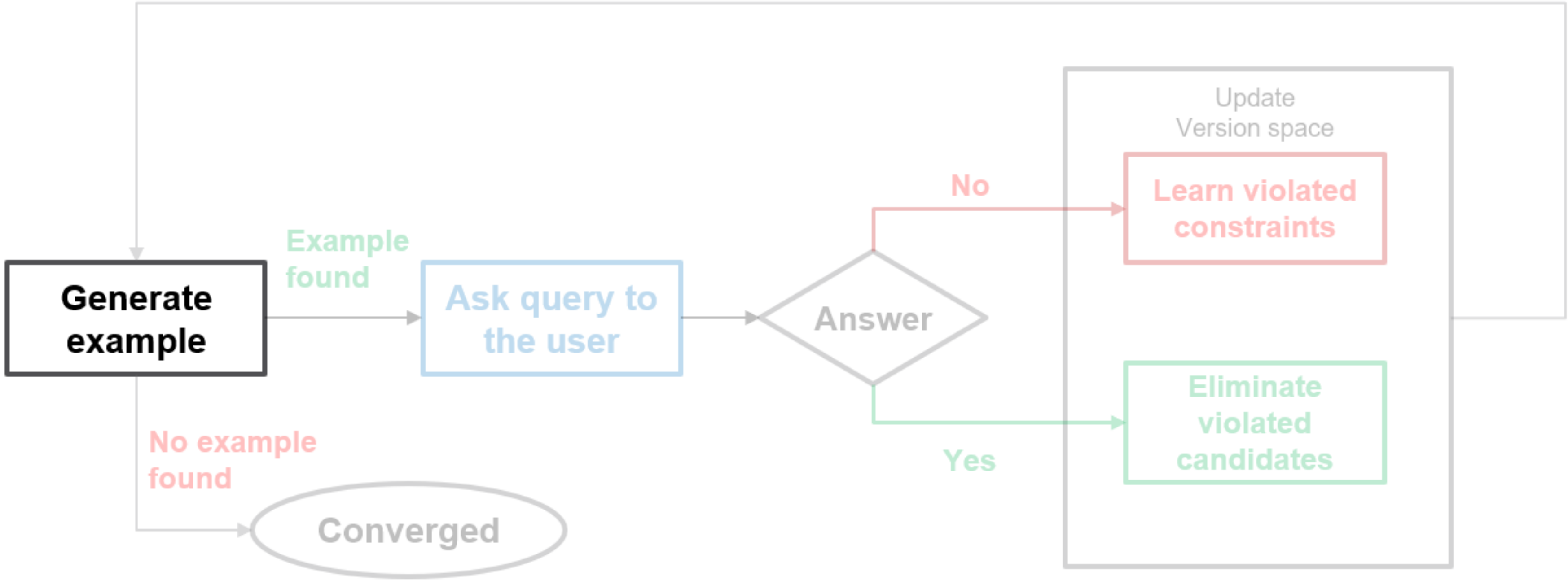
Query generation

Why??

Find an informative ("irredundant") query

- Not violating any learned constraint in C_L
- Violating at least one constraint from *B*

Find $e \in sol(C_L \wedge \bigvee_{c \in B} \sim c)$



Learning Constraints through Partial Queries, C. Bessiere et al., AIJ, 2023

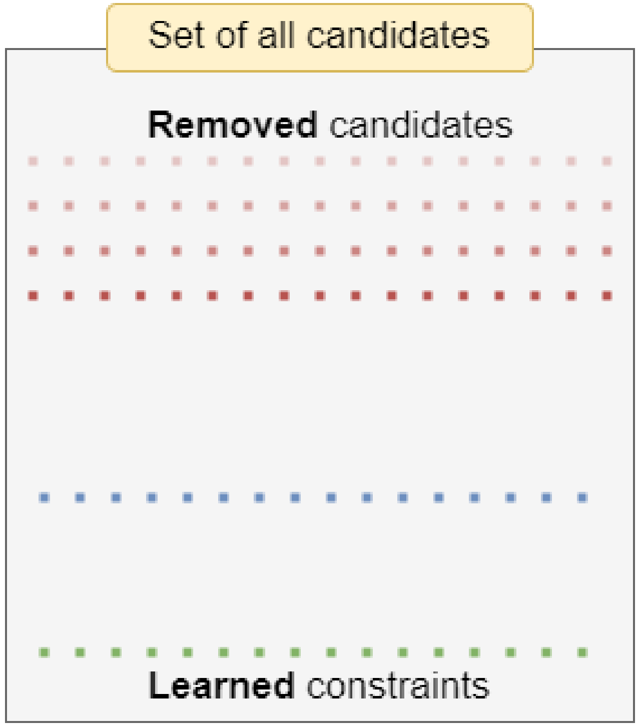
Adapting Candidate Elimination

- Examples: Assignments to the variables of the problem

- Learning from ***positive*** examples (Solutions):

- Violated constraints cannot be part of the model
- Otherwise, it could not be a solution

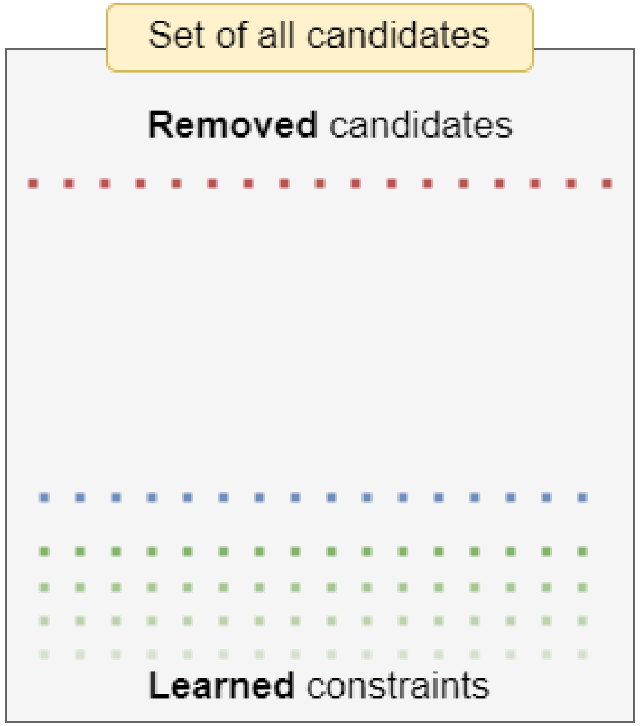
Shrinking the bias



- Learning from ***negative*** examples (Non-solutions):

- One (or more) violated constraint is a constraint of the problem
- Otherwise, it would be a solution

Learning Constraints

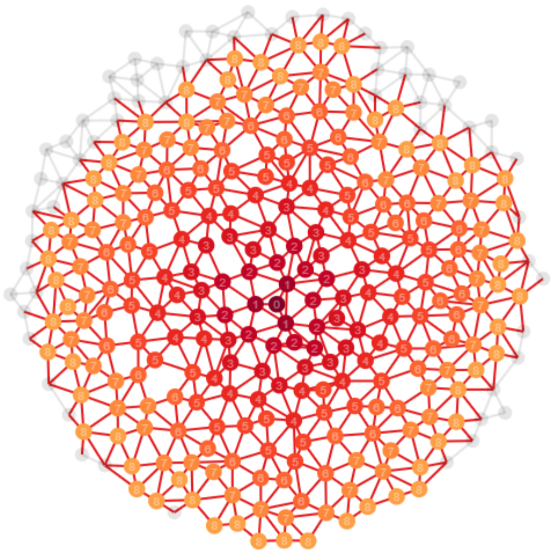


Query generation

Finding an informative query

- Not always an easy task …

B can be huge!!



B can contain indirectly implied constraints

Assume a simple 9x9 Sudoku puzzle.
- Combinations of ≠ constraints imply others
- 648 of them imply the rest 162

When the 648 constraints have been learned and must be satisfied, the rest cannot be violated!

Indirect implications are not detected with simple propagation!!

Query generation

Custom solvers

- Custom solvers are often employed to deal with this

Projection-based query generation (**PQ-Gen**)

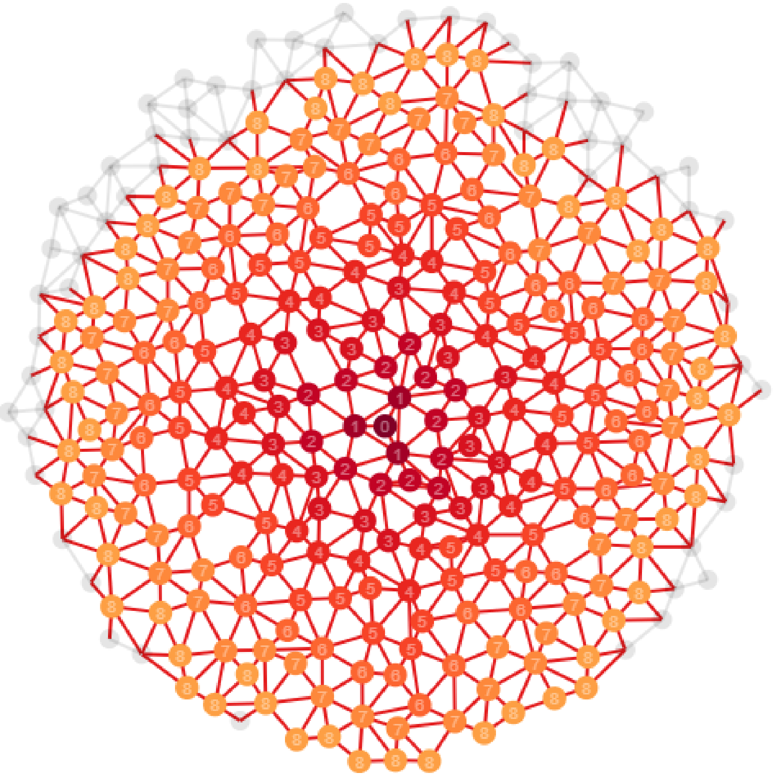
- Project down to the relevant variables

$$Y = \bigcup_{c \in B} var(c)$$

  - Either way, we can only get information on variables of constraints in B
  - Avoiding indirect implications

KU LEUVEN

What if the set of candidates is too large??

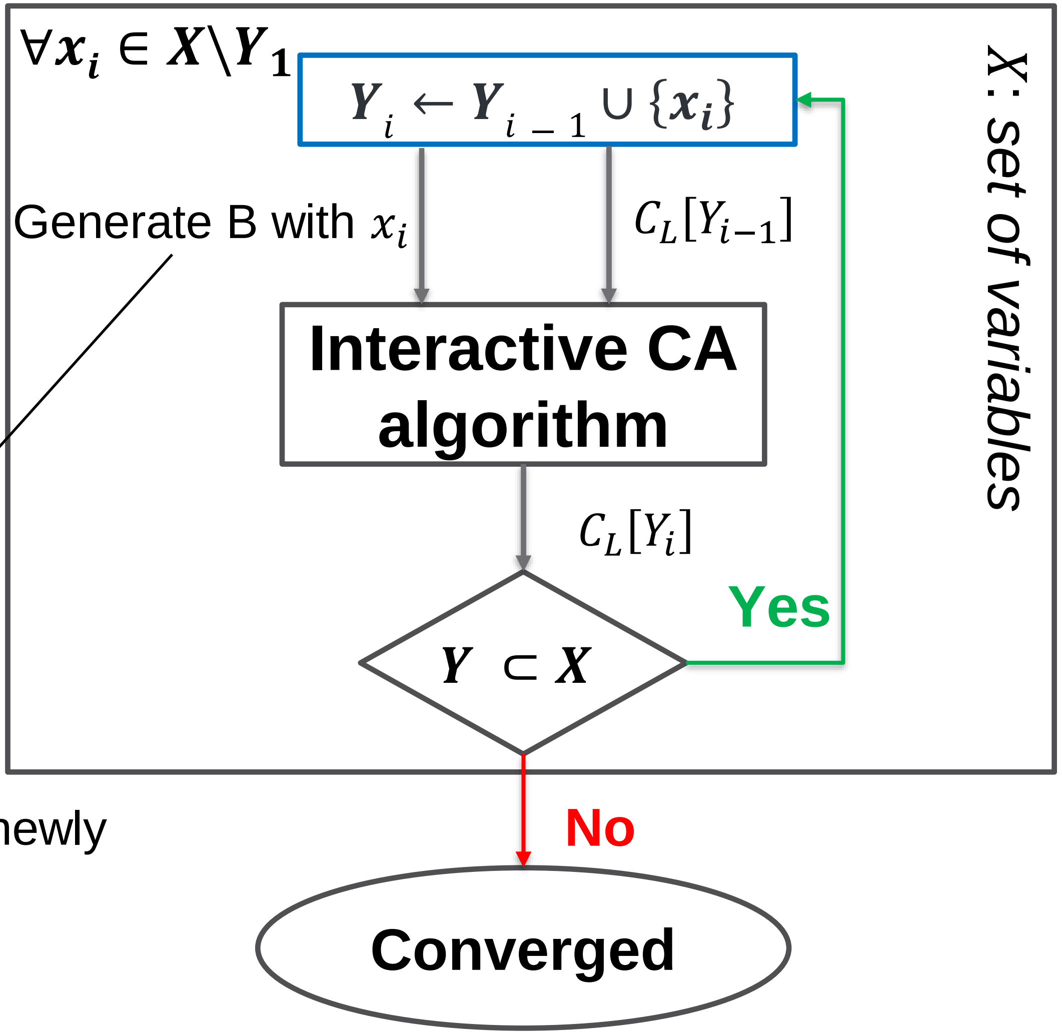


- Can’t store all of it at the same time??
- Too slow??

KU LEUVEN

GrowAcq: Growing Acquisition

Start with $Y_1 \leftarrow \emptyset$, or a small subset of X

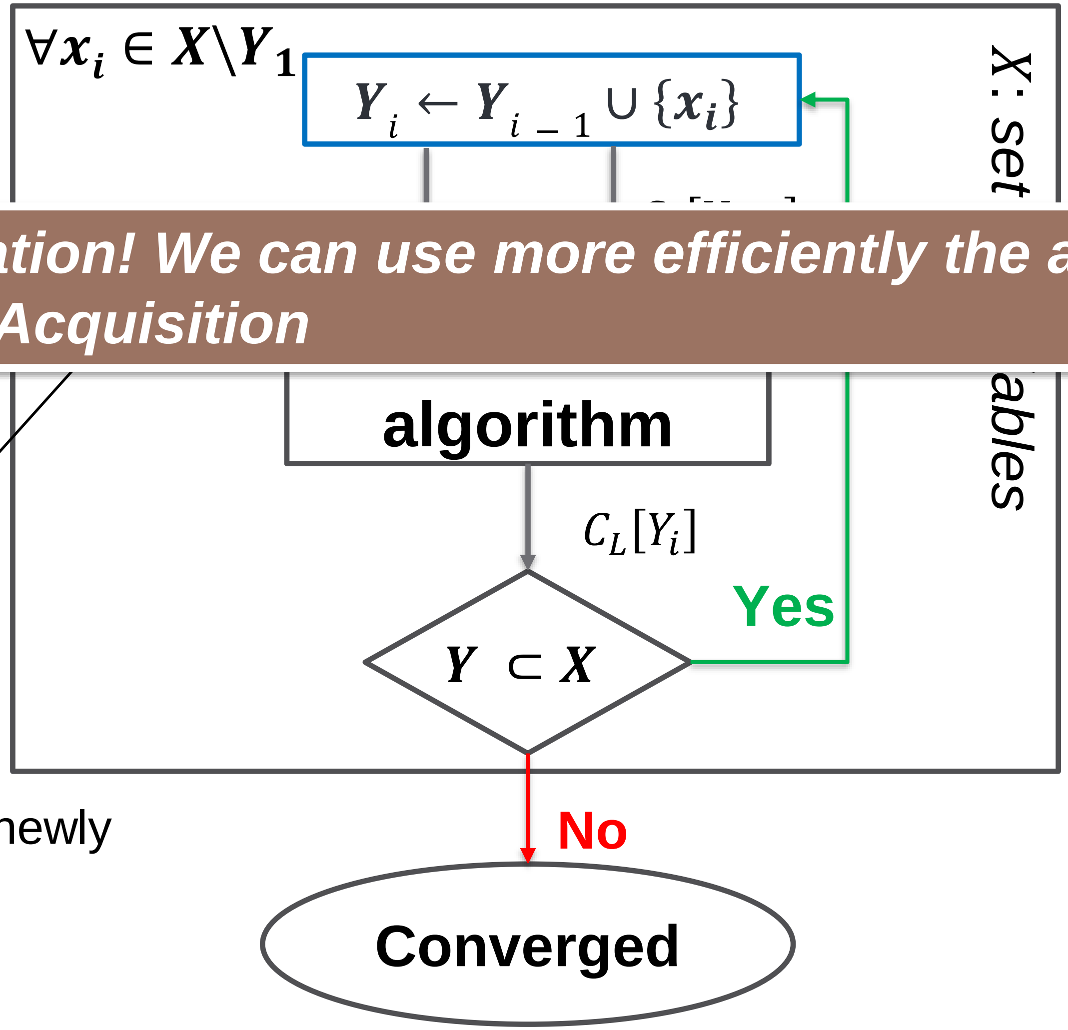


Bottom-up approach: Using Interactive CA algorithms, on an (incrementally growing) subset of variables

Need only constraints that newly added variable participates

GrowAcq: Growing Acquisition

Start with $Y_1 \leftarrow \emptyset$, or a small subset of X



Saves time during query generation! We can use more efficiently the available time to guide better Constraint Acquisition

Bot
Interactive CA algorithms, on an (incrementally growing) subset of variables

Need only constraints that newly added variable participates

Guided Bottom-up Constraint Acquisition, D. Tsouros et al., CP, 2023

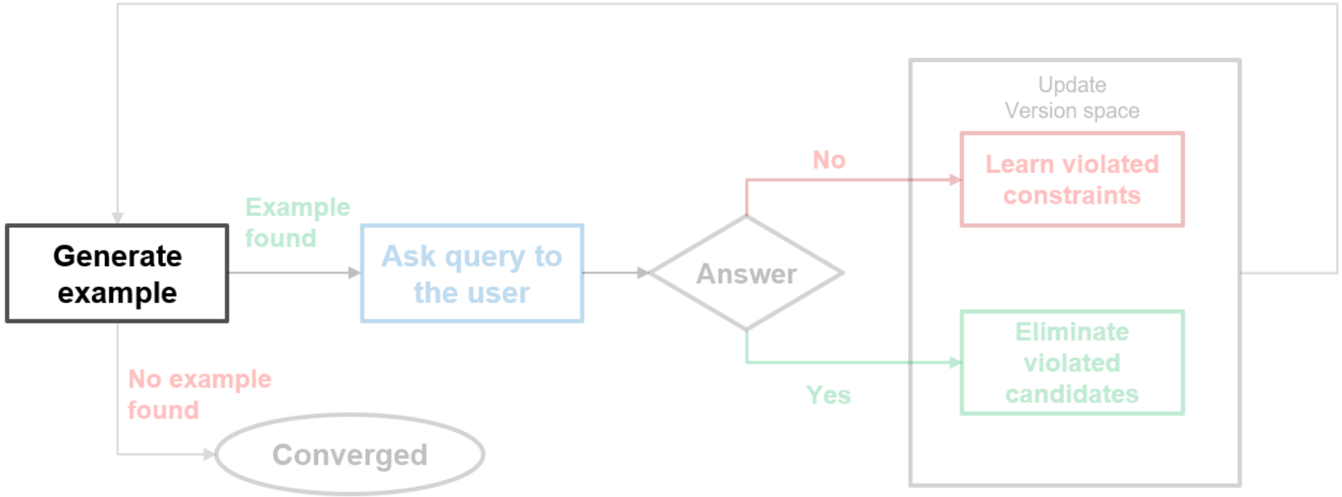
Query generation

Informative query

Quality of query

- Get the maximum amount of information

Convergence



Guiding Query Generation

Quality of query

- *Better* generated examples lead to faster convergence
  - More information per query -> less queries needed

Typically: maximize candidate violations

$$\max \sum_{c \in B} \sim c$$

Not fully aligning with the goal!!

Guiding Query Generation

Better generated examples lead to faster convergence

Positive answers: shrink *B* fast

- Negative answers: Find the conflict fast

The more we have violated the faster B will shrink

The less candidates we have violated, the less queries we need to find the constraint(s)

$$\boldsymbol{max} \sum_{\boldsymbol{c \in B}} [\![\boldsymbol{e} \notin \boldsymbol{sol(c)}]\!]$$

Opposite objectives based on the (future) answer

$$\boldsymbol{min} \sum_{\boldsymbol{c \in B}} [\![\boldsymbol{e} \notin \boldsymbol{sol(c)}]\!]$$

Guided Bottom-up Constraint Acquisition, D. Tsouros et al., CP, 2023

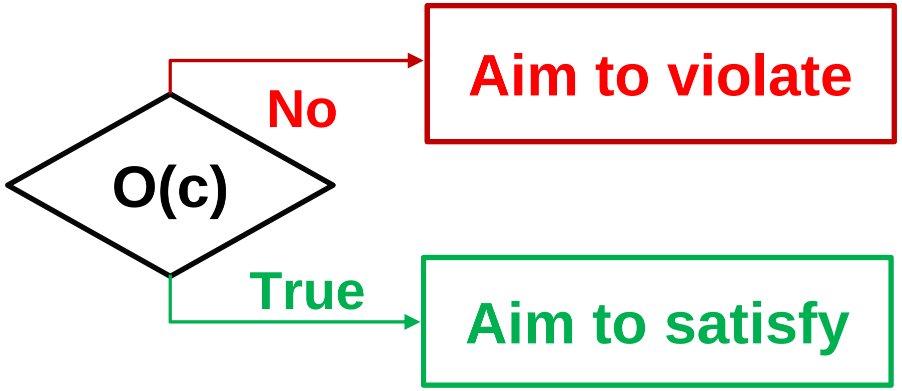
KU LEUVEN

Guiding Query Generation

- Positive answers: shrink *B* fast → $\boldsymbol{max} \sum_{c \in B} [\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!]$
- Negative answers: Find the conflict fast → $\boldsymbol{min} \sum_{c \in B} [\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!]$

What if we can predict if a candidate is a constraint of the problem or not?

Use of *Oracle* $\mathrm{O(c)} = (\mathrm{c} \in CT)$, to guide query generation based on the *prediction of the constraint*



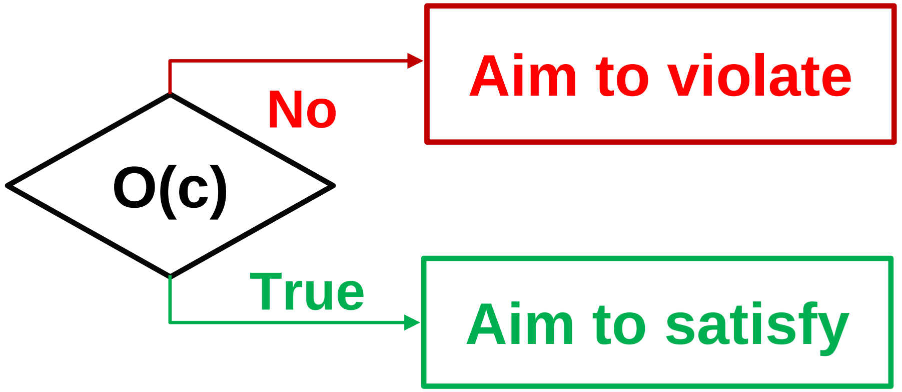
Why?

1. Aim for positive answers first: $\boldsymbol{max}(\sum_{c \in B} [\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!])$
2. When a (probably true) constraint has to be violated, leading to a *negative answer* $\boldsymbol{min}(\sum_{c \in B} [\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!])$

Guided Bottom-up Constraint Acquisition, D. Tsouros et al., CP, 2023

What can we use as the oracle $O(c)$ to guide CA?

The oracle $O(c)$ decides if we should satisfy or violate a candidate



- Using probabilities for the constraints,
- Take the decision that will minimize the (expected) number of queries

$$O(c) = \frac{1}{\log(|Y|)} \leq P(c)$$

Minimize the *expected* number of queries
$|Y|$: size of the example
$\log(|Y|)$: number of queries for each constraint when not guided
$\frac{1}{\log(|Y|)}$: Percentage of queries resulting on a constraint learnt

Probability Estimation

Any method to estimate probabilities can be exploited in this step

$$O(c) = \frac{1}{\log(|Y|)} \leq P(c)$$

We estimate P with a frequentist approach, by counting based on the relation rel(c) of the constraints

$\boldsymbol{N_{C_L}}$: Number of times a constraint with rel(c) was found to be part of C_T
$\boldsymbol{N_R}$: number of times a constraint with rel(c) was found to not be part of C_T

$$P(c) = \frac{N_{C_L}}{N_{C_L} + N_R}$$

KU LEUVEN

Experimental Evaluation

Questions:

[Q1] Performance of PQ-Gen

[Q2] Performance of GrowAcq

[Q3] Performance of our probability-guided objective function for query generation

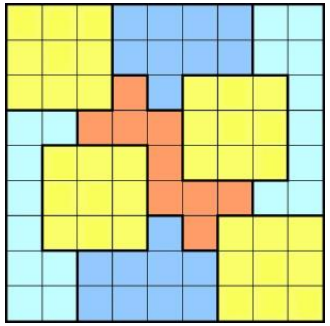
[Q4] Performance of the combination of our methods

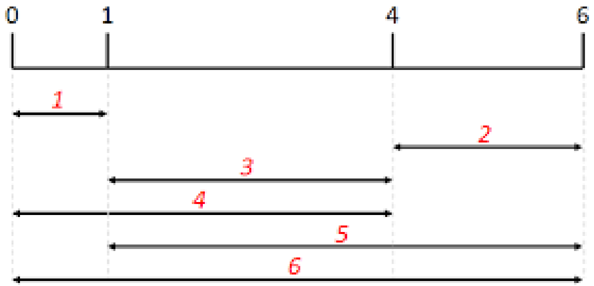
[Q5] Applying them on problems with huge sets of candidates (up to 1.5M)

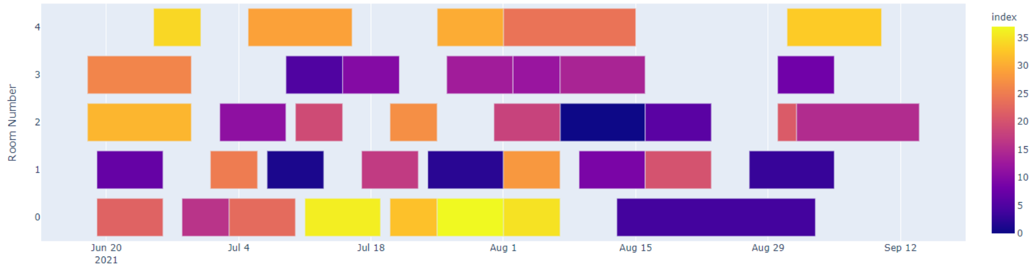
Compared:

- ***MQuAcq-2:*** *SOTA algorithm, baseline for the comparison*
- ***GrowAcq + MQuAcq-2****: Our bottom-up approach with MQuAcq-2*
- ***GrowAcq + MQuAcq-2 guided****:* Guiding using the proposed objective function

Benchmarks
- Implemented in CPMpy







Metrics*:*

- *# of Queries*
- *Max waiting Time (s)*
- *Total time (s)*

Lower is better

KU LEUVEN

Q1: Avoiding premature convergence

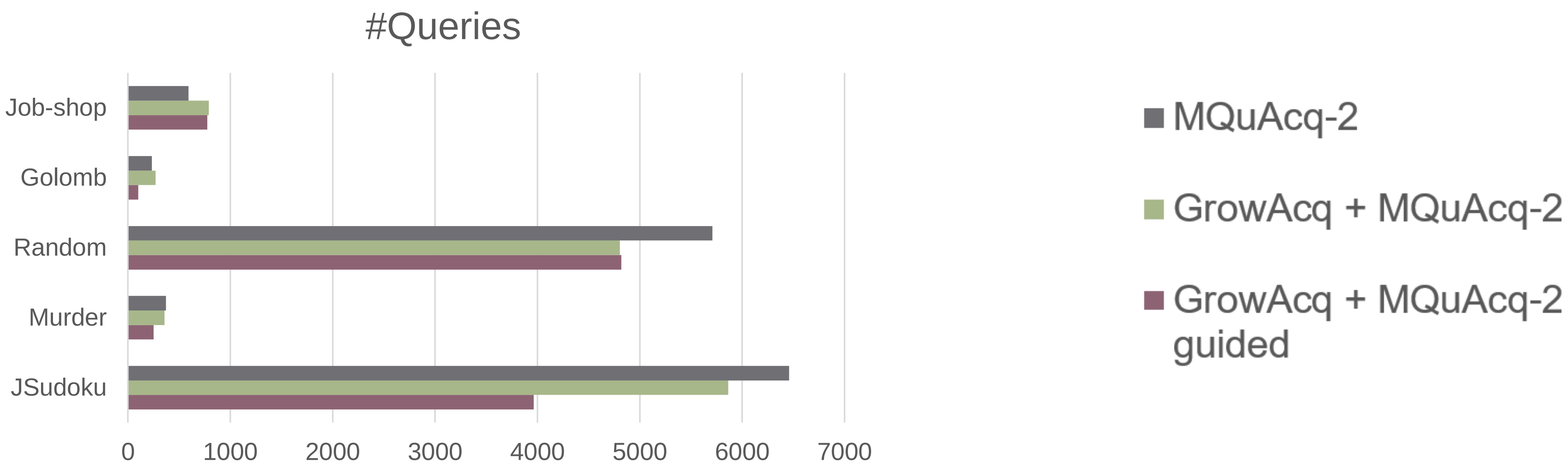
Methods using conventional solvers
TQ-Gen: Time-Bounded Query Generation
PQ-Gen: Projection-based Query Generation (proposed)

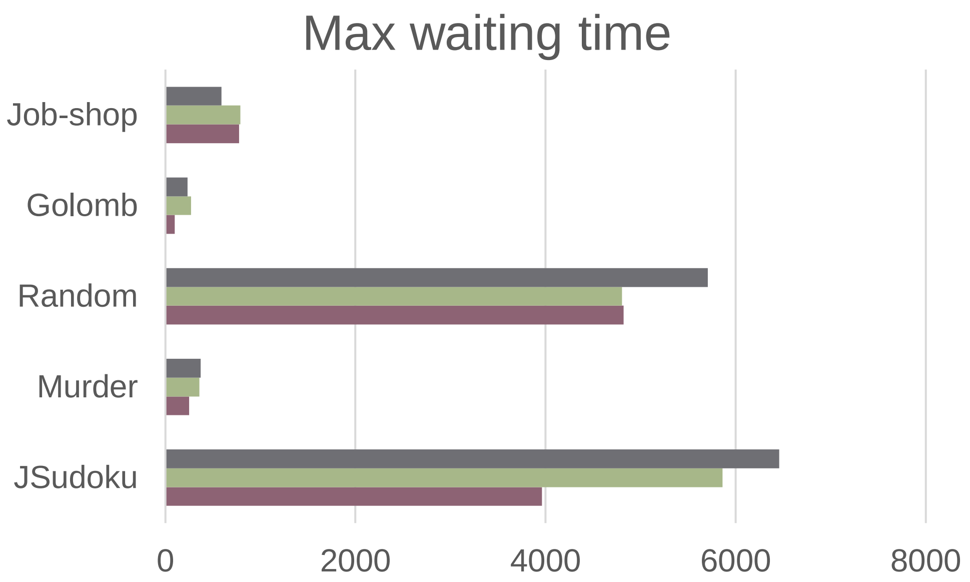
Evaluated using different configurations
Benchmark: Jigsaw Sudoku

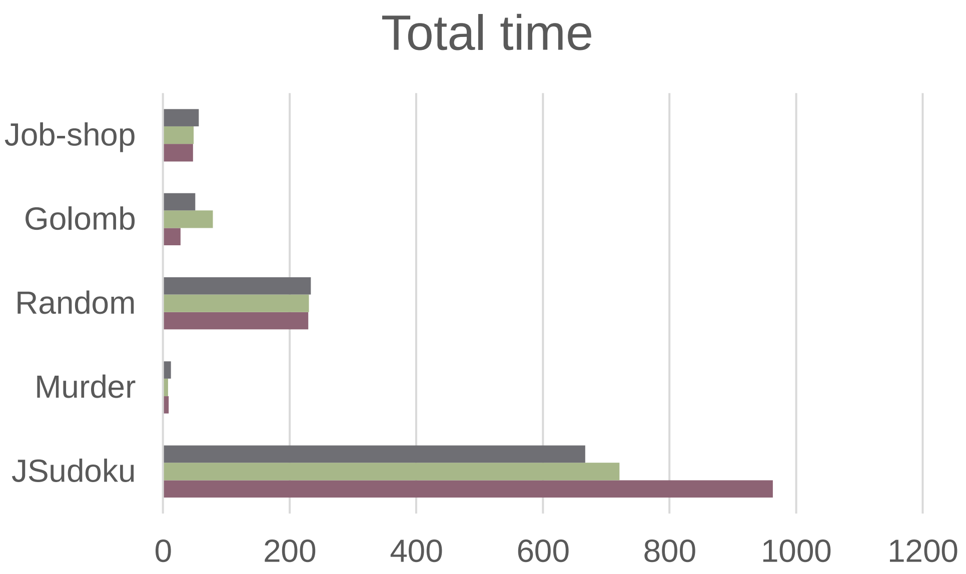
Method	$Conv$	$\#q$	T_{max}	T_{total}
MQuAcq-2 with TQ-Gen [1]	32%	7 555	20.66	2 371.40
MQuAcq-2 with PQ-Gen (ours)	100%	6 551	4.42	728.25

KU LEUVEN

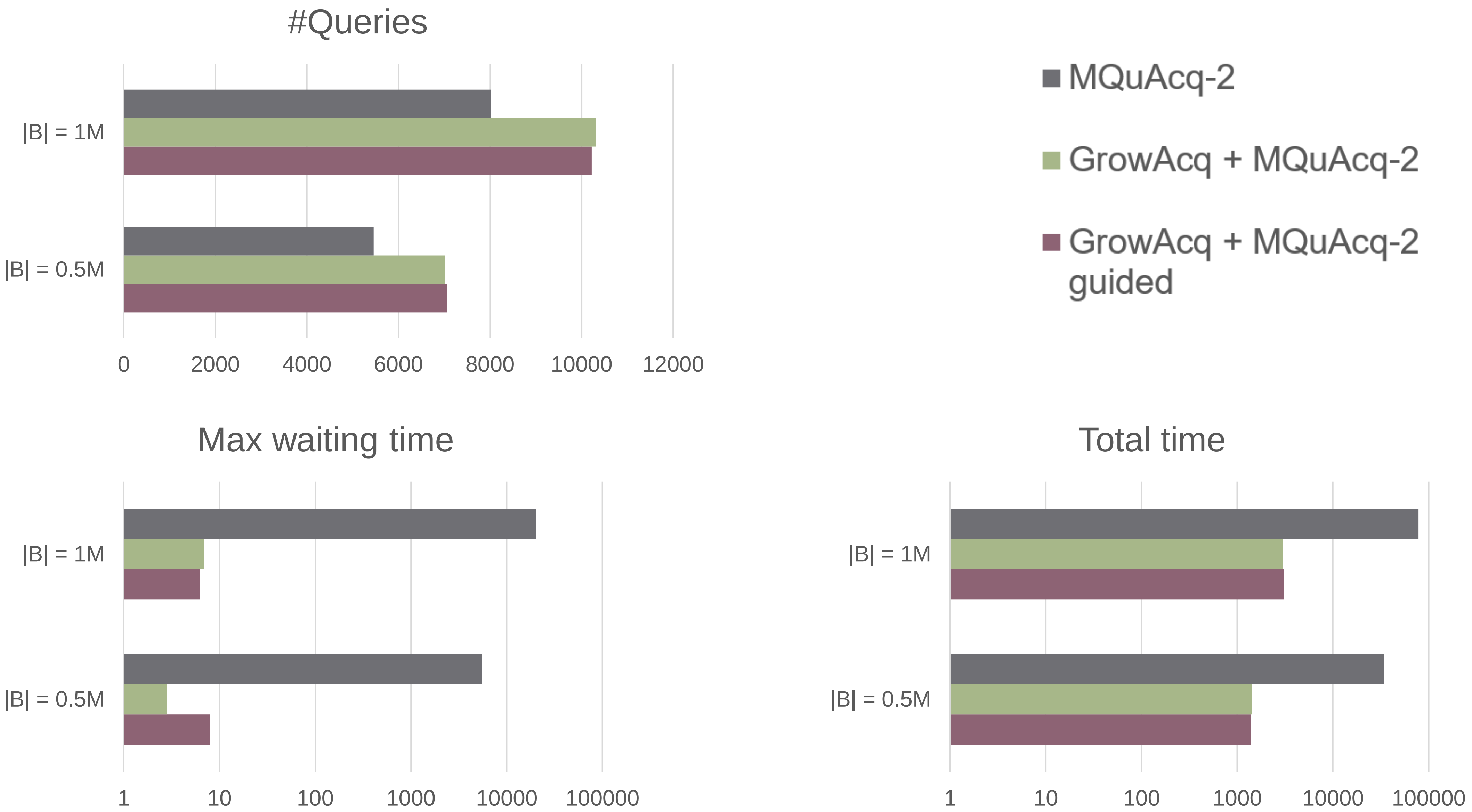
Q2-Q4: Performance of our methods



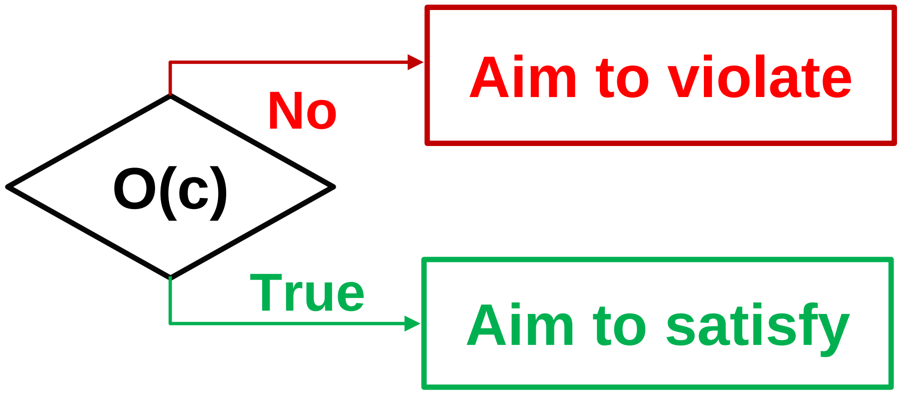




Q5: Dealing with larger sets of candidates

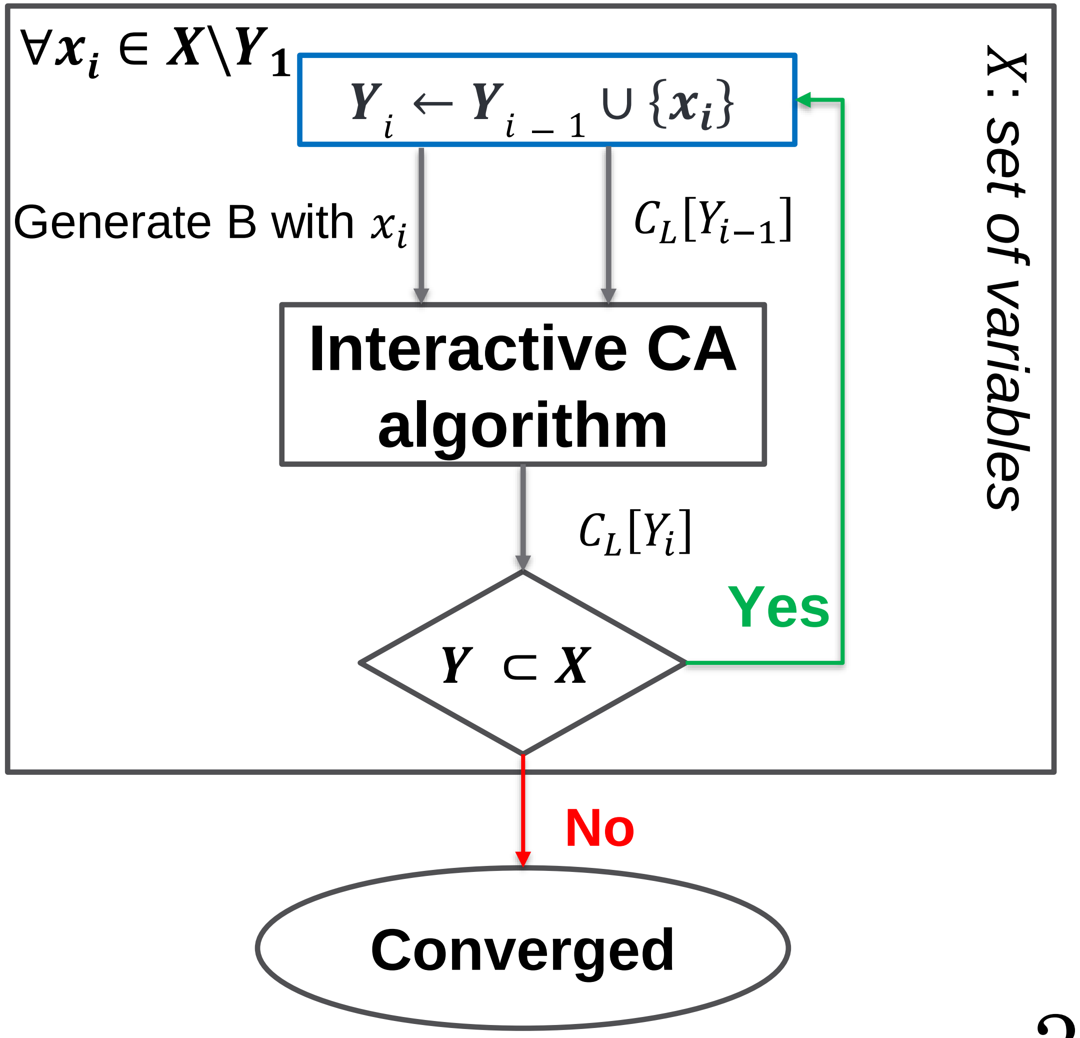


Conclusions



- PQ-Gen can be utilized to exploit conventional solvers in interactive CA
- Using GrowAcq's bottom-up procedure improves CA systems performance (especially when B is large)
- Guiding constraint acquisition using probabilities for the candidate constraints significantly reduces the number of queries

Start with $Y_1 \leftarrow \emptyset$, or a small subset of X



Github repository:
https://github.com/Dimosts/ActiveConLearn

KU LEUVEN

Challenges

Number of queries

- Number of queries needed to converge is still large.
- Guide the acquisition process using information from the learned constraint set

Specific classes of constraints

- Importantly: Global constraints

Noisy data

- Alleviate the assumption that the user can answer all the queries posted correctly

Thank you for your attention