Réduction de la dépendance aux connaissances préalables en acquisition de contraintes

Areski HIMEUR

Université de Montpellier, France

Soutenance de thèse – 03 décembre 2025

Devant le jury :

Gilles AUDEMARD	Professeur des universités, Université d'Artois – **Rapporteur**
Christian BESSIERE	Directeur de recherche, CNRS – **Directeur de thèse**
Clément CARBONNEL	Chargé de recherche, CNRS – **Co-encadrant de thèse**
Florence DUPIN DE SAINT-CYR	Maîtresse de conférences, Université de Toulouse – **Examinatrice**
Simon de GIVRY	Chargé de recherche, MIAT - INRAE Toulouse – **Rapporteur**
Marie-José HUGUET	Professeure des universités, INSA Toulouse - LAAS-CNRS – **Examinatrice**

What is Constraint Programming?

Constraint Programming (CP) is a paradigm for solving combinatorial problems.
CP involves two main steps:

1. **Modeling** the problem with constraints,
2. **Solving** with a generic solver.

What is Constraint Programming?

Constraint Programming (CP) is a paradigm for solving combinatorial problems.
CP involves two main steps:

1. **Modeling** the problem with constraints,
2. **Solving** with a generic solver.

Benefits:

- Separation of modeling and solving,
- Expressiveness for combinatorial problems,
- Efficient generic solving techniques.

What is Constraint Programming?

Constraint Programming (CP) is a paradigm for solving combinatorial problems.
CP involves two main steps:

1. **Modeling** the problem with constraints,
2. **Solving** with a generic solver.

Benefits:

- Separation of modeling and solving,
- Expressiveness for combinatorial problems,
- Efficient generic solving techniques.

CP is widely used in scheduling, logistics, timetabling, etc.

Constraint Programming Workflow

Planning a Work Schedule (Nurse Rostering Problem)

Imagine you need to assign shifts to 15 nurses over a week with 3 shifts per day while respecting the following rules:

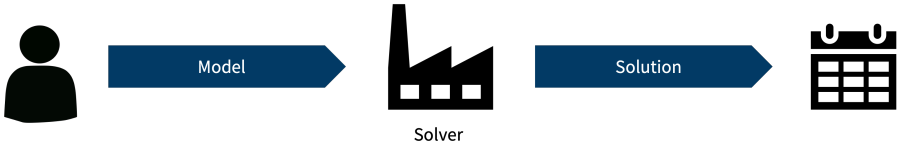
- Each shift must be covered by 3 nurses.
- A nurse cannot be assigned to two different shifts on the same day.
- A nurse cannot work the last shift of a day and the first shift of the next day.

Constraint Programming Workflow

Planning a Work Schedule (Nurse Rostering Problem)

Imagine you need to assign shifts to 15 nurses over a week with 3 shifts per day while respecting the following rules:

- Each shift must be covered by 3 nurses.
- A nurse cannot be assigned to two different shifts on the same day.
- A nurse cannot work the last shift of a day and the first shift of the next day.



Variables and Domains

Definition | Variable

A **variable** represents an unknown that needs to be determined.

- A variable has an associated **domain**: a finite set of possible values it can take.

Variables and Domains

Definition | Variable

A **variable** represents an unknown that needs to be determined.

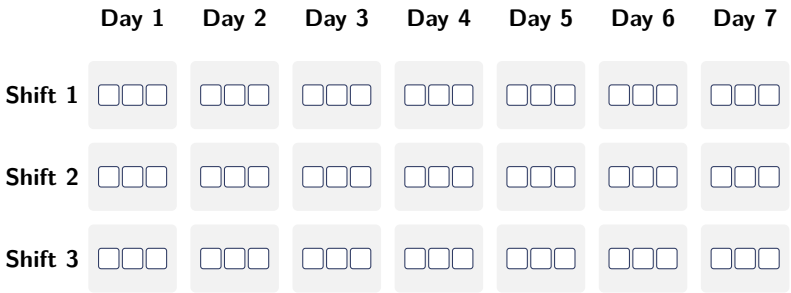
- A variable has an associated **domain**: a finite set of possible values it can take.

An **assignment** gives a value from the domain to each variable.

Example

In the nurse rostering problem:

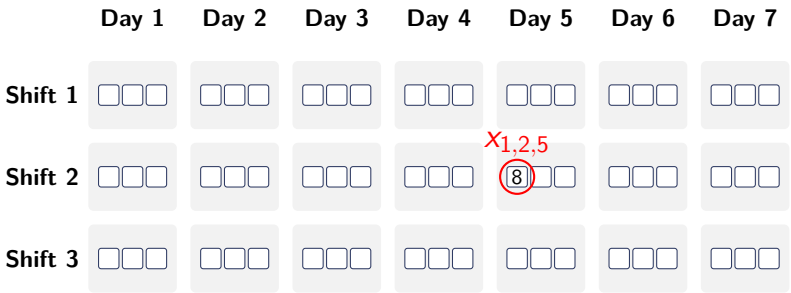
- Variables X: $x_{i,j,k}$ is the nurse assigned to slot i of shift j on day k.
- Domain: $\{1, 2, \ldots, 15\}$ (available nurses)



Example

In the nurse rostering problem:

- Variables X: $x_{i,j,k}$ is the nurse assigned to slot i of shift j on day k.
- Domain: $\{1, 2, \ldots, 15\}$ (available nurses)



Constraint

Definition | Constraint

A **constraint** over a set of variables X and a domain D is a pair (S, R) where:

- $S \subseteq X$ is the scope (the variables involved).
- R is the relation (the allowed combinations of values for the variables in S).

Constraint

Definition | Constraint

A **constraint** over a set of variables X and a domain D is a pair (S, R) where:

- $S \subseteq X$ is the scope (the variables involved).
- R is the relation (the allowed combinations of values for the variables in S).

A constraint **accepts** an assignment if values in S belong to R; otherwise it **rejects** it.

Constraint

Definition | Constraint

A **constraint** over a set of variables X and a domain D is a pair (S, R) where:

- $S \subseteq X$ is the scope (the variables involved).
- R is the relation (the allowed combinations of values for the variables in S).

A constraint **accepts** an assignment if values in S belong to R; otherwise it **rejects** it.
The **arity** of a constraint is the number of variables in its scope.

Definition | Constraint

A **constraint** over a set of variables X and a domain D is a pair (S, R) where:

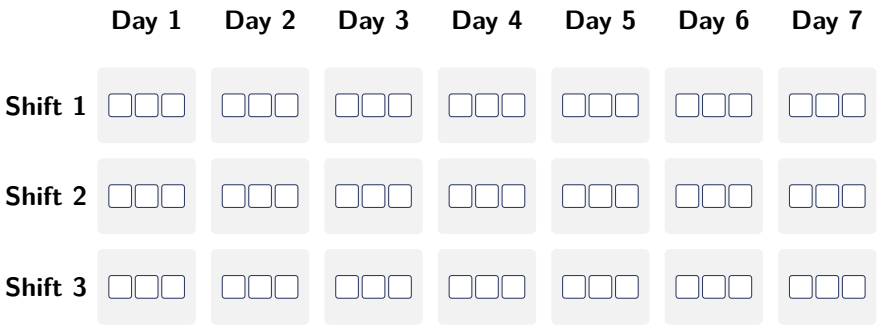
- $S \subseteq X$ is the scope (the variables involved).
- R is the relation (the allowed combinations of values for the variables in S).

A constraint **accepts** an assignment if values in S belong to R; otherwise it **rejects** it.
The **arity** of a constraint is the number of variables in its scope.

Example

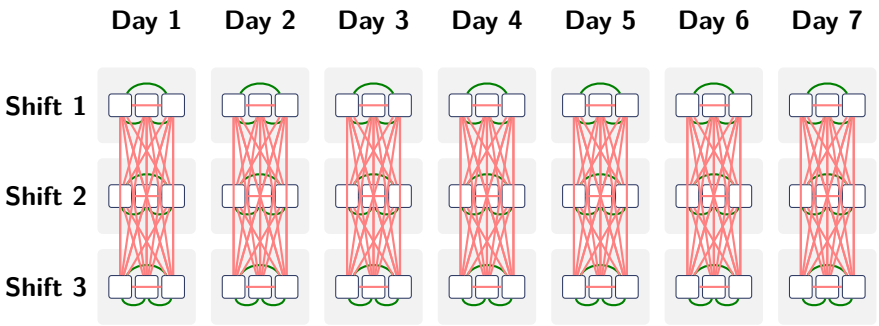
- $S = (x_{1,2,5}, x_{2,2,5})$
- $R = \{(a, b) \mid a \neq b\}$

The Constraints of the Nurse Rostering Problem

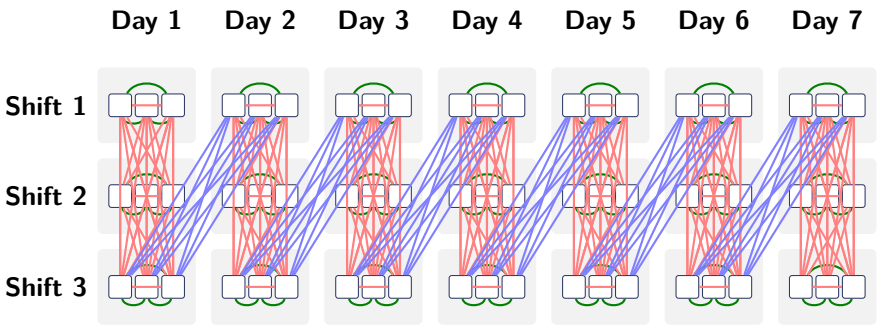


The Constraints of the Nurse Rostering Problem

The Constraints of the Nurse Rostering Problem



The Constraints of the Nurse Rostering Problem



Definition | Constraint Network

A **constraint network** is a triplet $N = (X, D, C)$ where:

- X is a set of **variables** $\{x_1, \ldots, x_n\}$;
- D is a finite **domain** of values for the variables;
- C is a set of **constraints** $\{c_1, \ldots, c_m\}$.

Definition | Constraint Network

A **constraint network** is a triplet $N = (X, D, C)$ where:

- X is a set of **variables** $\{x_1, \ldots, x_n\}$;
- D is a finite **domain** of values for the variables;
- C is a set of **constraints** $\{c_1, \ldots, c_m\}$.

Definition | Constraint Language

A **constraint language** Γ is a set of relations over a domain.

- A network N is over Γ if all constraints of N use relations from Γ.

Define the problem as a constraint network

Variables and Domains:

- Variables X: $x_{i,j,k}$ is the nurse assigned to slot i of shift j on day k.
- Domain D: $\{1, 2, \ldots, 15\}$.

Define the problem as a constraint network

Variables and Domains:

- Variables X: $x_{i,j,k}$ is the nurse assigned to slot i of shift j on day k.
- Domain D: $\{1, 2, \ldots, 15\}$.

Constraints:

- Each shift must be covered by at least 3 nurses.
  - $x_{i,j,k} \neq x_{i',j,k}$ for all $i \neq i'$, shifts j and days k.
- A nurse cannot be assigned to two different shifts on the same day.
  - $x_{i,j,k} \neq x_{i,j',k}$ for all slots k and k', shifts $j \neq j'$ for all days k.
- A nurse cannot work the last shift of a day and the first shift of the next day.
  - $x_{i,3,k} \neq x_{i',1,k+1}$ for all slots i and i' and days k.

Define the problem as a constraint network

Variables and Domains:

- Variables X: $x_{i,j,k}$ is the nurse assigned to slot i of shift j on day k.
- Domain D: $\{1, 2, \ldots, 15\}$.

Constraints:

- Each shift must be covered by at least 3 nurses.
  - $x_{i,j,k} \neq x_{i',j,k}$ for all $i \neq i'$, shifts j and days k.
- A nurse cannot be assigned to two different shifts on the same day.
  - $x_{i,j,k} \neq x_{i,j',k}$ for all slots k and k', shifts $j \neq j'$ for all days k.
- A nurse cannot work the last shift of a day and the first shift of the next day.
  - $x_{i,3,k} \neq x_{i',1,k+1}$ for all slots i and i' and days k.

This network is defined over the constraint language $\Gamma = \{\neq\}$.

Define the problem as a constraint network

Variables and Domains:

- Variables X: $x_{i,j,k}$ is the nurse assigned to slot i of shift j on day k.
- Domain D: $\{1, 2, \ldots, 15\}$.

Constraints:

- Each shift must be covered by at least 3 nurses.
  - $x_{i,j,k} \neq x_{i',j,k}$ for all $i \neq i'$, shifts j and days k.
- A nurse cannot be assigned to two different shifts on the same day.
  - $x_{i,j,k} \neq x_{i,j',k}$ for all slots k and k', shifts $j \neq j'$ for all days k.
- A nurse cannot work the last shift of a day and the first shift of the next day.
  - $x_{i,3,k} \neq x_{i',1,k+1}$ for all slots i and i' and days k.

This network is defined over the constraint language $\Gamma = \{\neq\}$.

Challenge | Designing a constraint network representing a given problem can be difficult.

Examples and Consistency

Definition | Example

An example over a set of variables X with domain D is composed of:

- an assignment to X;
- a classification label as a *positive example* or as a *negative example*.

Definition | Consistency

A constraint network N is consistent with a set of examples E if:

- All positive examples in E are solutions of N.
- All negative examples in E are non-solutions of N.

Illustration of Passive Constraint Acquisition

Formal Definition

Definition | Passive Constraint Acquisition Task

Input:
- (X, D): a set of variables and a finite domain;
- Γ: a constraint language;
- E: a set of examples.

Goal: Find a constraint network over Γ consistent with E.

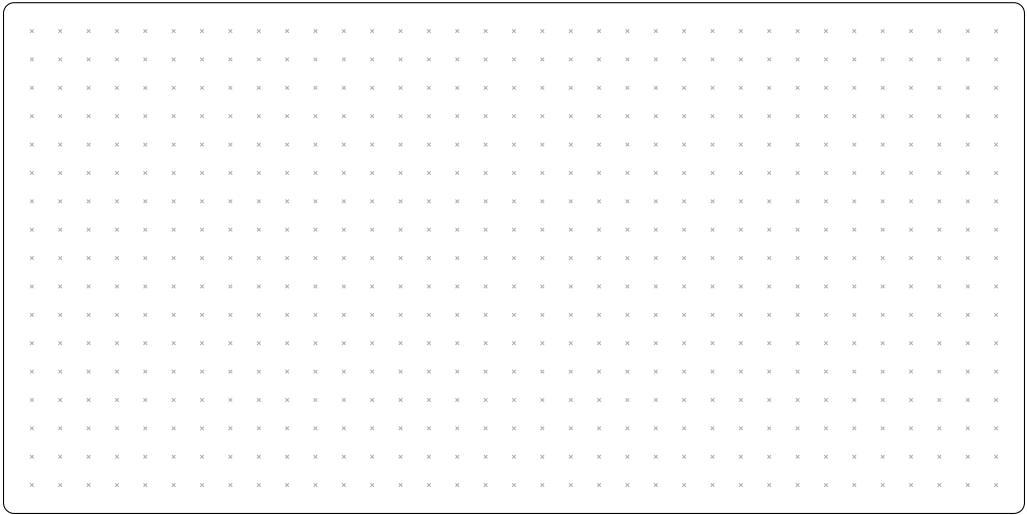
Formal Definition

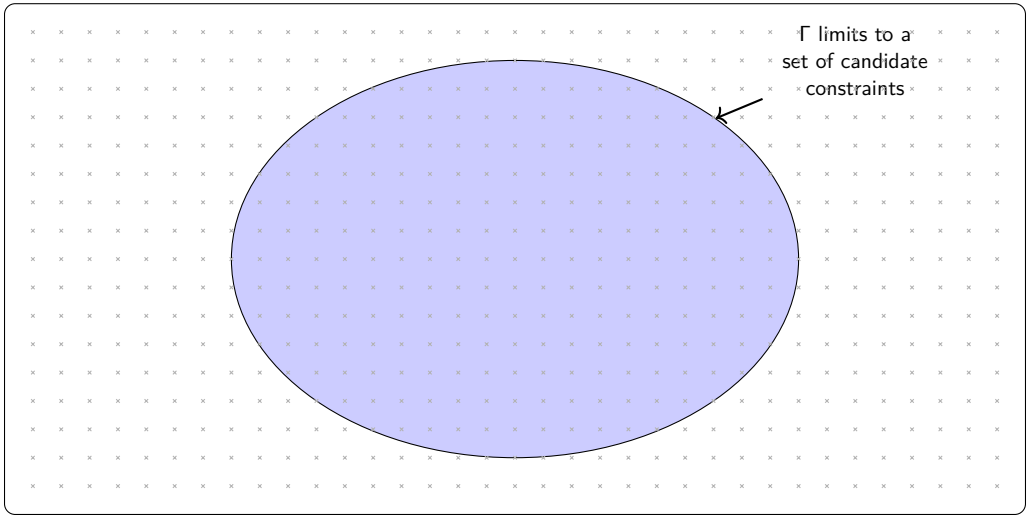
Definition | Passive Constraint Acquisition Task

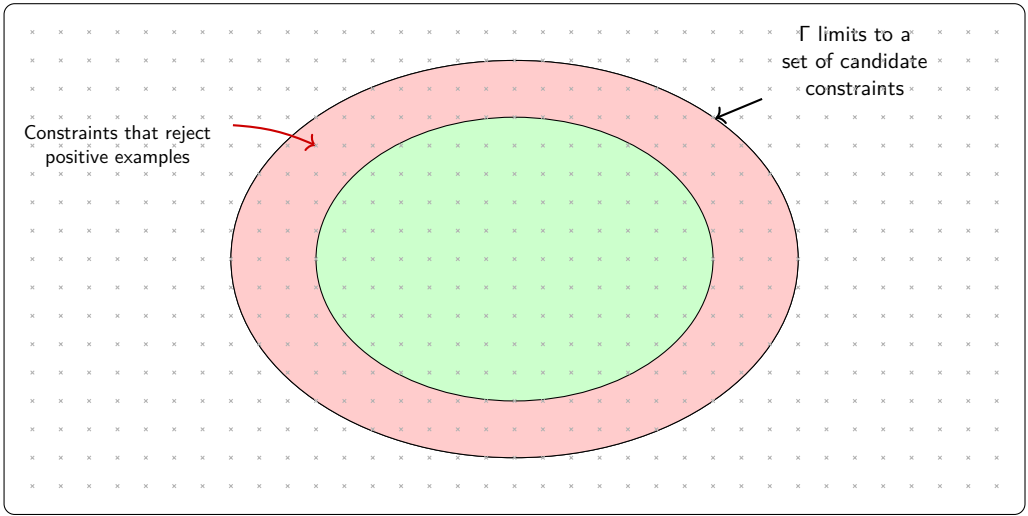
Input:
- (X, D): a set of variables and a finite domain;
- Γ: a constraint language;
- E: a set of examples.

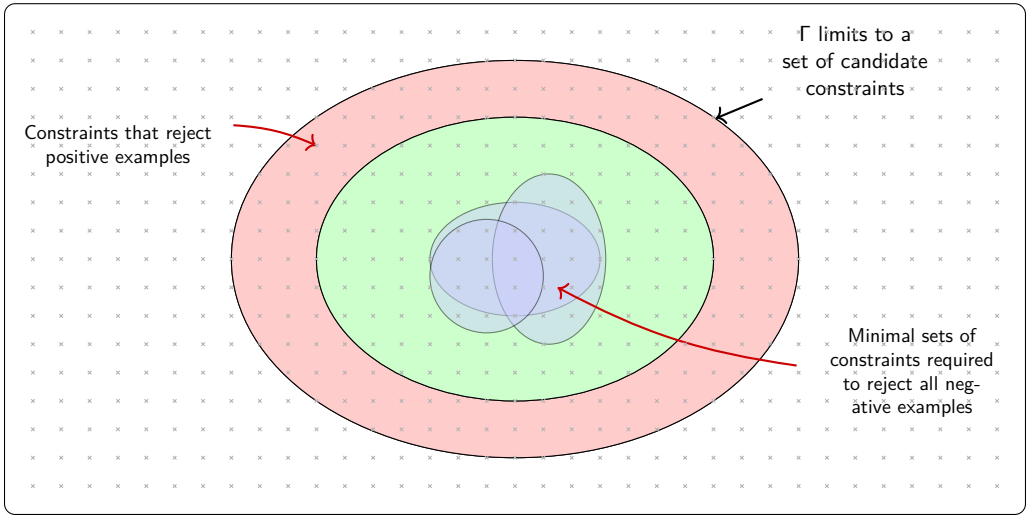
Goal: Find a constraint network over Γ consistent with E.

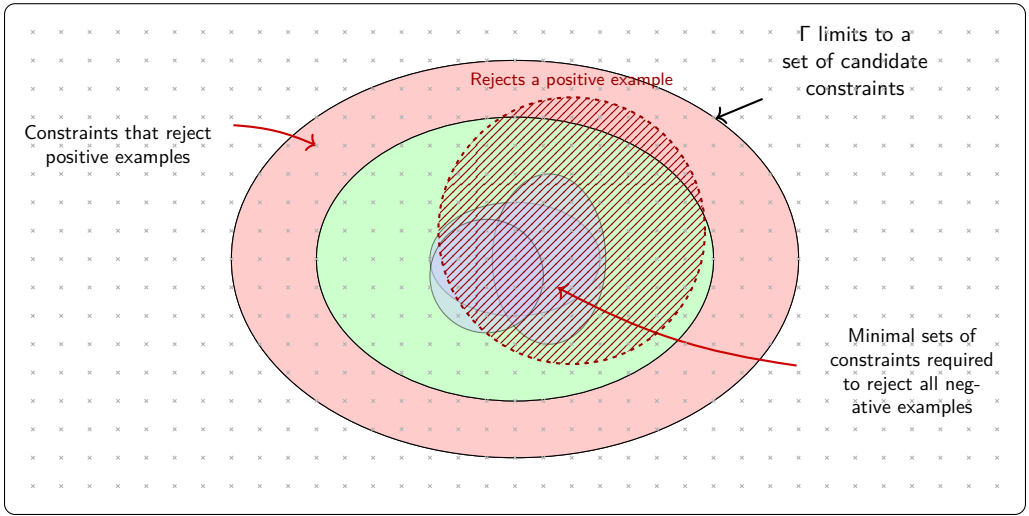
ConAcq.1 [Bessiere et al., 2005, 2017], ModelSeeker [Beldiceanu and Simonis, 2012], BayesAcq [Prestwich et al., 2021], Count-CP [Kumar et al., 2022]

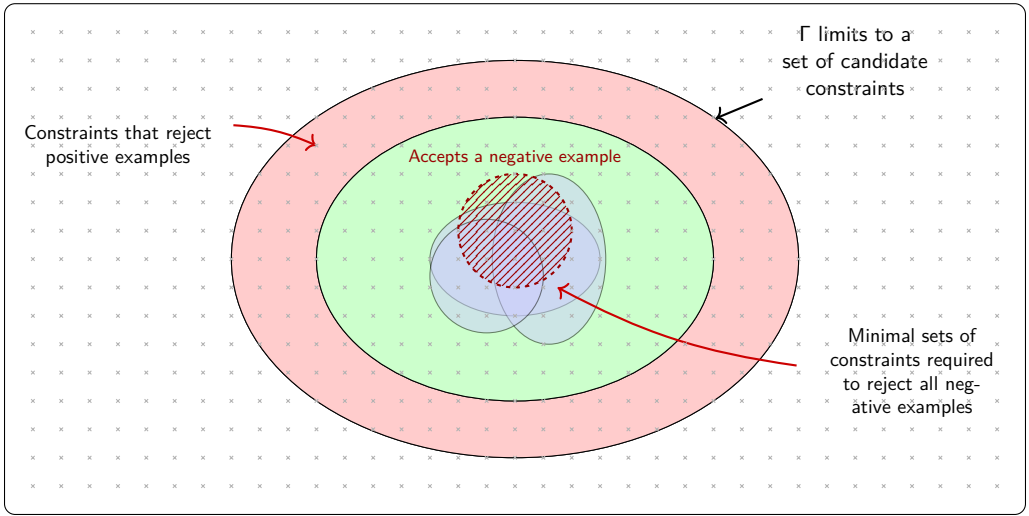


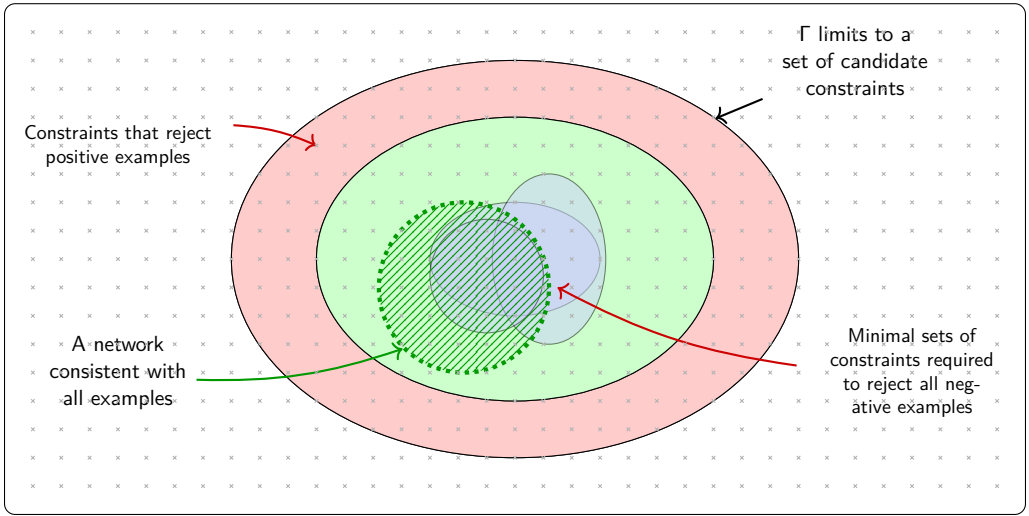

Γ limits to a
set of candidate
constraints

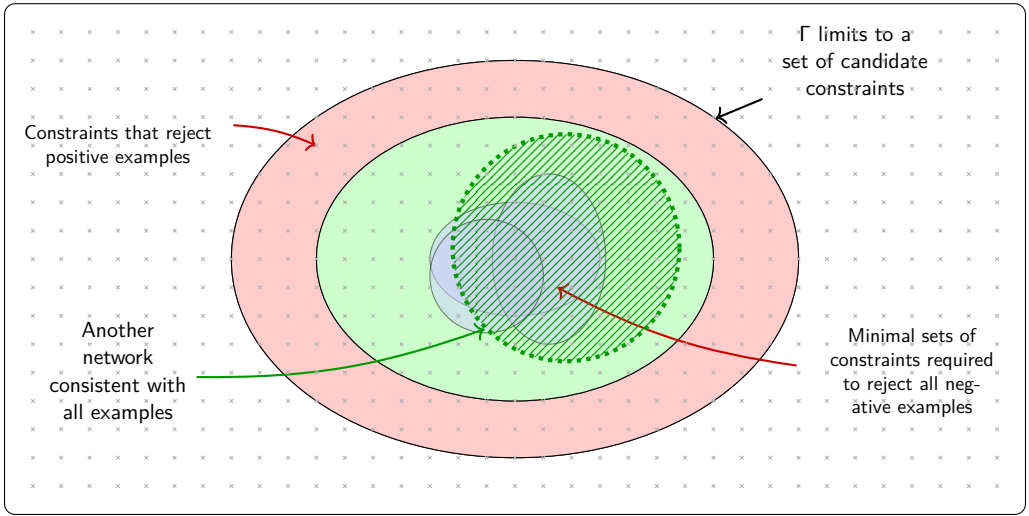

Γ limits to a
set of candidate
constraints
Constraints that reject
positive examples


Γ limits to a set of candidate constraints
Constraints that reject positive examples
Minimal sets of constraints required to reject all negative examples


Γ limits to a set of candidate constraints
Rejects a positive example
Constraints that reject positive examples
Minimal sets of constraints required to reject all negative examples


Γ limits to a set of candidate constraints
Constraints that reject positive examples
Accepts a negative example
Minimal sets of constraints required to reject all negative examples


Γ limits to a set of candidate constraints
Constraints that reject positive examples
A network consistent with all examples
Minimal sets of constraints required to reject all negative examples


Γ limits to a set of candidate constraints
Constraints that reject positive examples
Another network consistent with all examples
Minimal sets of constraints required to reject all negative examples

Definition | Passive Constraint Acquisition Task

Input:
- (X, D): a set of variables and a finite domain;
- Γ: a constraint language;
- E: a set of examples.

Goal: Find a constraint network over Γ consistent with E.

Definition | Passive Constraint Acquisition Task

Input:
- (X, D): a set of variables and a finite domain;
- Γ: a constraint language;
- E: a set of examples.

Goal: Find a constraint network over Γ consistent with E.

Definition | Passive Constraint Acquisition Task

Input:
- (X, D): a set of variables and a finite domain;
- Γ: a constraint language;
- E: a set of examples.

Goal: Find a constraint network over Γ consistent with E.

How to choose the constraint language?

Definition | Passive Constraint Acquisition Task

Input:
- (X, D): a set of variables and a finite domain;
- Γ: a constraint language;
- E: a set of examples.

Goal: Find a constraint network over Γ consistent with E.

How to choose the constraint language?

Problem | Reliance on prior knowledge prevents automated modeling.

Motivation

All current approaches require some knowledge of the constraint language of the output network.

Our contribution

We develop a constraint acquisition method that
constructs a constraint language as part of the learning process.

In general, given a set of examples, a large number of constraint languages can be used.

In general, given a set of examples, a large number of constraint languages can be used.

Problem | Some languages are clearly unsatisfactory from a practical point of view

In general, given a set of examples, a large number of constraint languages can be used.

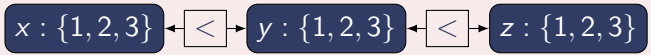
Problem | Some languages are clearly unsatisfactory from a practical point of view

- Examples:
  - $(x = 1, y = 2, z = 3)$ is a positive example.
  - $(x = 3, y = 2, z = 1)$, $(x = 1, y = 3, z = 2)$ and $(x = 3, y = 1, z = 2)$ are negative examples.

In general, given a set of examples, a large number of constraint languages can be used.

Problem | Some languages are clearly unsatisfactory from a practical point of view

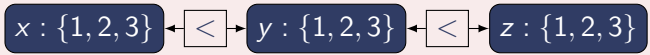
- Examples:
  - $(x = 1, y = 2, z = 3)$ is a positive example.
  - $(x = 3, y = 2, z = 1)$, $(x = 1, y = 3, z = 2)$ and $(x = 3, y = 1, z = 2)$ are negative examples.
- A consistent network:



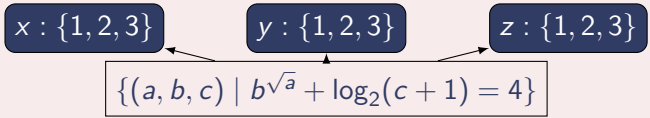
In general, given a set of examples, a large number of constraint languages can be used.

Problem | Some languages are clearly unsatisfactory from a practical point of view

- Examples:
  - $(x = 1, y = 2, z = 3)$ is a positive example.
  - $(x = 3, y = 2, z = 1)$, $(x = 1, y = 3, z = 2)$ and $(x = 3, y = 1, z = 2)$ are negative examples.
- A consistent network:



- Another consistent network:



Intuition | The best constraint language is the simplest.

First approximation: the **smallest** language in terms of its arity and number of relations.

Intuition | The best constraint language is the simplest.

First approximation: the **smallest** language in terms of its arity and number of relations.

Sub-Problem

Instance: Set of examples E; two integers k and r.

Question: Is there a constraint network **over a language with at most k relations and arity at most r** and consistent with E?

Intuition | The best constraint language is the simplest.

First approximation: the **smallest** language in terms of its arity and number of relations.

Sub-Problem

Instance: Set of examples E; two integers k and r.

Question: Is there a constraint network **over a language with at most k relations and arity at most r** and consistent with E?

This problem is NP-complete even for $(k, r) = (1, 1)$.

The LFA method (Language-Free Acquisition)

Method | Compute a consistent constraint network with minimum (k, r).

- **Strategy:** minimize $k + r^2$

The LFA method (Language-Free Acquisition)

Method | Compute a consistent constraint network with minimum (k, r).

- **Strategy:** minimize $k + r^2$
- **Tie-breaking:** lower arity, more constraints, tighter constraints

The LFA method (Language-Free Acquisition)

Method | Compute a consistent constraint network with minimum (k, r).

- **Strategy:** minimize $k + r^2$
- **Tie-breaking:** lower arity, more constraints, tighter constraints

Construct and solve a model for each (k, r) by increasing order.

Find a network for a given (k, r)

For given (k, r), we compute a constraint network or prove that none exists by solving a Weighted Partial Max-Sat instance.

Find a network for a given (k, r)

For given (k, r), we compute a constraint network or prove that none exists by solving a Weighted Partial Max-Sat instance.

- **We ensure that the constraint language has k relations and arity r.**

Find a network for a given (k, r)

For given (k, r), we compute a constraint network or prove that none exists by solving a Weighted Partial Max-Sat instance.

- **We ensure that the constraint language has k relations and arity r.**
- **We ensure that the output network is consistent with the examples.**

Find a network for a given (k, r)

For given (k, r), we compute a constraint network or prove that none exists by solving a Weighted Partial Max-Sat instance.

- **We ensure that the constraint language has k relations and arity r.**
- **We ensure that the output network is consistent with the examples.**
- **We optimise the following criteria:**
  - First we maximize the number of constraints.
  - Then we maximize the tightness of constraints.

Find a network for a given (k, r)

For given (k, r), we compute a constraint network or prove that none exists by solving a Weighted Partial Max-Sat instance.

- **We ensure that the constraint language has k relations and arity r.**
- **We ensure that the output network is consistent with the examples.**
- **We optimise the following criteria:**
  - First we maximize the number of constraints.
  - Then we maximize the tightness of constraints.

$$\text{Try } (k = 1, r = 1) \xrightarrow{\textit{fail}} (k = 2, r = 1) \xrightarrow{\textit{fail}} \cdots \xrightarrow{\textit{success}} \text{Output Network}$$

Protocol for experiments

Examples Generation

- We define a **target network** for various benchmarks.

Protocol for experiments

Examples Generation

- We define a **target network** for various benchmarks.
- Generate examples from the target with **50% positive** and **50% negative** examples.

Protocol for experiments

Examples Generation

- We define a **target network** for various benchmarks.
- Generate examples from the target with **50% positive** and **50% negative** examples.

Run LFA to learn a network with different numbers of examples.

Protocol for experiments

Examples Generation

- We define a **target network** for various benchmarks.
- Generate examples from the target with **50% positive** and **50% negative** examples.

Run LFA to learn a network with different numbers of examples.

Evaluation

- **Accuracy:** Count examples needed for **100% accuracy** (independent test set).

Protocol for experiments

Examples Generation

- We define a **target network** for various benchmarks.
- Generate examples from the target with **50% positive** and **50% negative** examples.

Run LFA to learn a network with different numbers of examples.

Evaluation

- **Accuracy:** Count examples needed for **100% accuracy** (independent test set).
- **Qualitative:** Check if the learned network is:
  - the **target network**,
  - an **equivalent network**,
  - over the **target language**.

Summary of results over various benchmarks

Benchmark	Average number of examples for 100% accuracy	Language	Network
Sudoku	200	✓	✓
Jigsaw [*3 instances*]	900	✓	●
Nurse Rostering [*3 instances*]	467	✓	✓
Exam Timetabling [*3 instances*]	867	✓	✓
Schur's Lemma	600	✓	✓
Subgraph Isomorphism	700	✗	●
Golomb Ruler (10 variables)	3200	✗	●
8-Queens (coordinates model)	-	✗	✗

✓ Target ● Equivalent ✗ Not learned

Summary of results over various benchmarks

Benchmark	Average number of examples for 100% accuracy	Language	Network
Sudoku	200	✓	✓
Jigsaw [*3 instances*]	900	✓	●
Nurse Rostering [*3 instances*]	467	✓	✓
Exam Timetabling [*3 instances*]	867	✓	✓
Schur's Lemma	600	✓	✓
Subgraph Isomorphism	700	✗	●
Golomb Ruler (10 variables)	3200	✗	●
8-Queens (coordinates model)	-	✗	✗

✓ Target ● Equivalent ✗ Not learned

Summary of results over various benchmarks

Benchmark	Average number of examples for 100% accuracy	Language	Network
Sudoku	200	✓	✓
Jigsaw [*3 instances*]	900	✓	●
Nurse Rostering [*3 instances*]	467	✓	✓
Exam Timetabling [*3 instances*]	867	✓	✓
Schur's Lemma	600	✓	✓
Subgraph Isomorphism	700	✗	●
Golomb Ruler (10 variables)	3200	✗	●
8-Queens (coordinates model)	-	✗	✗

✓ Target ● Equivalent ✗ Not learned

Summary of results over various benchmarks

Benchmark	Average number of examples for 100% accuracy	Language	Network
Sudoku	200	✓	✓
Jigsaw [*3 instances*]	900	✓	●
Nurse Rostering [*3 instances*]	467	✓	✓
Exam Timetabling [*3 instances*]	867	✓	✓
Schur's Lemma	600	✓	✓
Subgraph Isomorphism	700	✗	●
Golomb Ruler (10 variables)	3200	✗	●
8-Queens (coordinates model)	-	✗	✗

✓ Target ● Equivalent ✗ Not learned

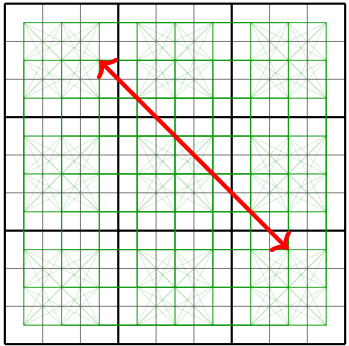
Summary of results over various benchmarks

Benchmark	Average number of examples for 100% accuracy	Language	Network
Sudoku	200	✓	✓
Jigsaw [*3 instances*]	900	✓	●
Nurse Rostering [*3 instances*]	467	✓	✓
Exam Timetabling [*3 instances*]	867	✓	✓
Schur's Lemma	600	✓	✓
Subgraph Isomorphism	700	✗	●
Golomb Ruler (10 variables)	3200	✗	●
8-Queens (coordinates model)	-	✗	✗

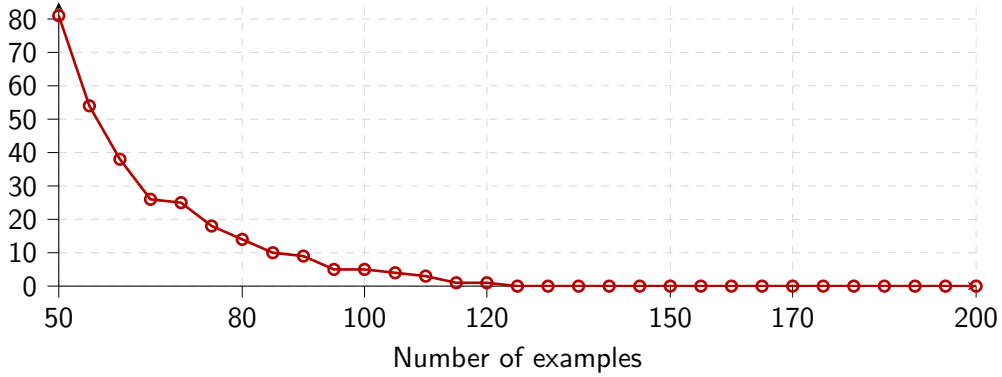
✓ Target ● Equivalent ✗ Not learned

Residual Constraints

Residual constraints are those that are not in the target network but are consistent with all provided examples.

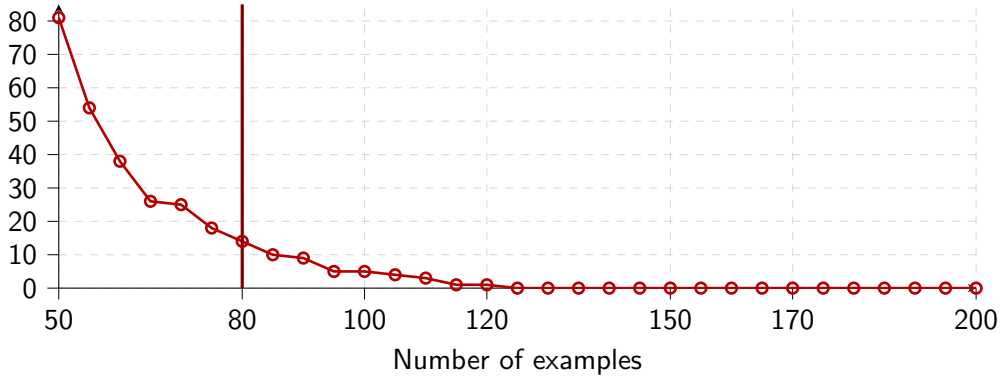


Residual Constraints



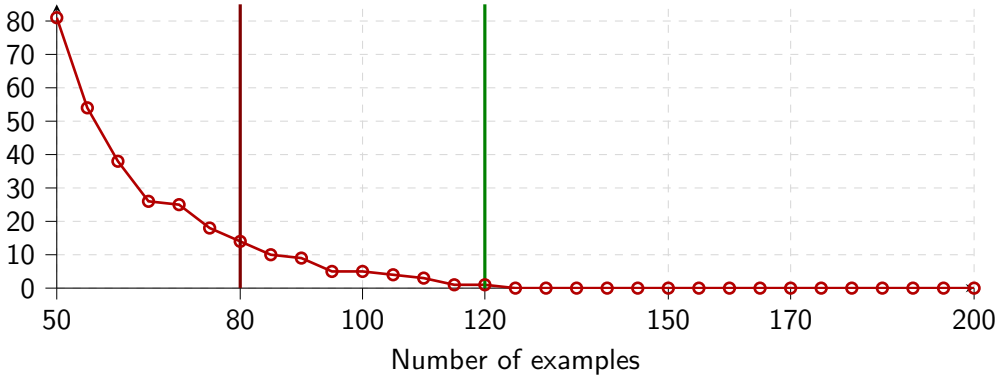
Number of residual constraints learned
as a function of the number of examples (Sudoku)

Residual Constraints



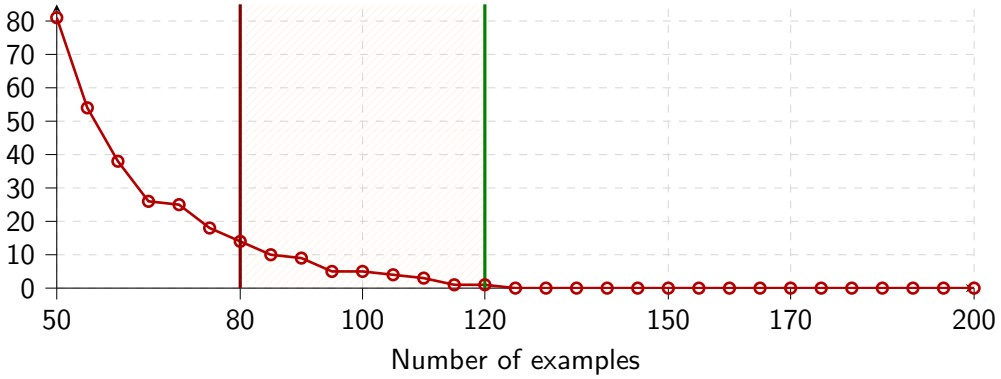
Number of residual constraints learned
as a function of the number of examples (Sudoku)

Residual Constraints



Number of residual constraints learned
as a function of the number of examples (Sudoku)

Residual Constraints



Number of residual constraints learned
as a function of the number of examples (Sudoku)

Limitation: Lack of Structure

Limitations

- LFA does not try to capture the structure of the problem;
- LFA requires numerous examples to eliminate few residual constraints.

Limitation: Lack of Structure

Limitations

- LFA does not try to capture the structure of the problem;
- LFA requires numerous examples to eliminate few residual constraints.

Goal

Instead of learning constraints using a flat list of scopes:

$$x_{1,1} \neq x_{1,2},\ x_{1,2} \neq x_{1,3},\ \ldots$$

we want to learn rules such as:

"All variables in the same row must be different."

Learning Compact Representations

Our claim

Learning compact representations of constraint networks is the key to better generalization.

Learning Compact Representations

Our claim

Learning compact representations of constraint networks is the key to better generalization.

To this end, we propose:

Learning Compact Representations

Our claim

Learning compact representations of constraint networks is the key to better generalization.

To this end, we propose:

1. A novel, compact representation for structured networks, which we call *template*.

Learning Compact Representations

Our claim

Learning compact representations of constraint networks is the key to better generalization.

To this end, we propose:

1. A novel, compact representation for structured networks, which we call *template*.
2. A new acquisition framework, TACQ, that learns these templates directly from examples.

What is a Template?

What is a Template?

1. **Attributes:** Functions $\{\phi_1, \phi_2, \ldots\}$ that assign numerical features to variables.

What is a Template?

1. **Attributes:** Functions $\{\phi_1, \phi_2, \ldots\}$ that assign numerical features to variables.

2. **Rules:** Mechanisms for producing many constraints based on attributes.
   - A relation (e.g., $\neq$),
   - A condition on some attributes of the variables (e.g., same attribute value).

   A rule produces a constraint for each scope of variables satisfying the condition.

Example | Template for the Sudoku

Variables: 81 variables $\{x_{i,j} \mid i, j \in [0..8]\}$

Attributes: $\phi_{\text{row}}(x_{i,j}) = i$; $\phi_{\text{col}}(x_{i,j}) = j$; $\phi_{\text{square}}(x_{i,j}) = \left\lfloor \frac{i}{3} \right\rfloor \times 3 + \left\lfloor \frac{j}{3} \right\rfloor$

ϕ_{row} ϕ_{col} ϕ_{square}

Rules:

1. Apply $\neq$ to (x, y) *if* $\phi_{\text{row}}(x) = \phi_{\text{row}}(y)$.
2. Apply $\neq$ to (x, y) *if* $\phi_{\text{col}}(x) = \phi_{\text{col}}(y)$.
3. Apply $\neq$ to (x, y) *if* $\phi_{\text{square}}(x) = \phi_{\text{square}}(y)$.

The TAcq Learning Framework

A two-step process:

The TAcq Learning Framework

A two-step process:

1. **Learn an initial network:** Use a baseline method to learn an initial network N

The TAcq Learning Framework

A two-step process:

1. **Learn an initial network:** Use a baseline method to learn an initial network N
2. **Refine this network into a template:** Learn a template that produces a large subset of the constraints of N.

The Template Learning Algorithm

Algorithm sketch

Input: A set of examples E and an initial network N consistent with E.

Output: A template T consistent with E.

The Template Learning Algorithm

Algorithm sketch

Input: A set of examples E and an initial network N consistent with E.

Output: A template T consistent with E.

---

1. Start with an empty template T

The Template Learning Algorithm

Algorithm sketch

Input: A set of examples E and an initial network N consistent with E.

Output: A template T consistent with E.

---

1. Start with an empty template T
2. **While** the template T is not consistent with E:

The Template Learning Algorithm

Algorithm sketch

Input: A set of examples E and an initial network N consistent with E.

Output: A template T consistent with E.

---

1. Start with an empty template T
2. **While** the template T is not consistent with E:
    1. **Guess a new attribute**

The Template Learning Algorithm

Algorithm sketch

Input: A set of examples E and an initial network N consistent with E.

Output: A template T consistent with E.

---

1. Start with an empty template T
2. **While** the template T is not consistent with E:
   1. **Guess a new attribute**
   2. **Greedily add rules that produce many new constraints of** N

Attribute Width and Generalization

The number of distinct values an attribute takes (its **width**) affects the maximum number of constraints produced by a rule based on that attribute (the **coverage**).

Width too small

Underfitting

Few constraints are produced.
We fail to capture the problem.

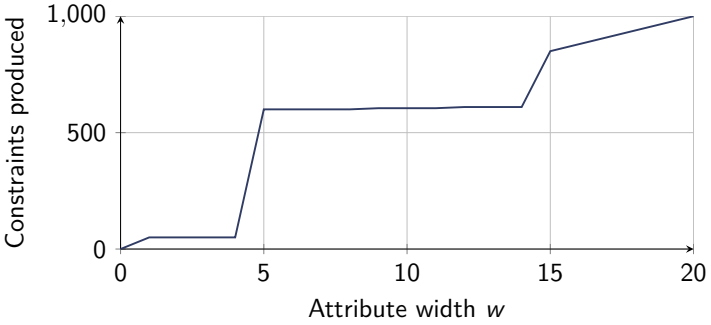
Width too large

Overfitting

Too many constraints are produced.
We capture the residual constraints.

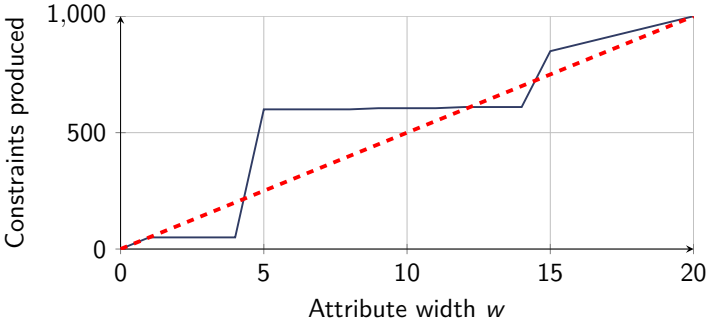
The Maximum Cover Above Expectation (MCAE) heuristic

The MCAE heuristic looks for the attribute with a trade-off between width and coverage.



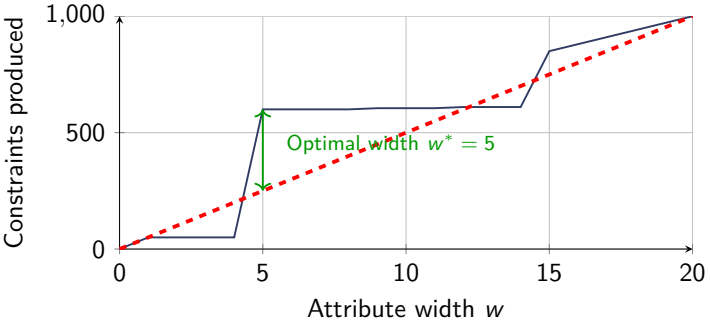
The Maximum Cover Above Expectation (MCAE) heuristic

The MCAE heuristic looks for the attribute with a trade-off between width and coverage.



The Maximum Cover Above Expectation (MCAE) heuristic

The MCAE heuristic looks for the attribute with a trade-off between width and coverage.



Protocol for experiments

We compare how many examples LFA and $\mathrm{LFA}+\mathrm{TA}\textsc{cq}$ need to learn benchmark problems.

Protocol for experiments

We compare how many examples LFA and $\mathrm{LFA}+\mathrm{TA}\textsc{cq}$ need to learn benchmark problems.

Protocol

Protocol for experiments

We compare how many examples LFA and $\mathrm{LFA}+\mathrm{TA}\textsc{cq}$ need to learn benchmark problems.

Protocol

- We generate an ordered sequence of examples,

Protocol for experiments

We compare how many examples LFA and $\mathrm{LFA}+\mathrm{TA}\textsc{cq}$ need to learn benchmark problems.

Protocol

- We generate an ordered sequence of examples,
- Both methods learn from the same sequence,

Protocol for experiments

We compare how many examples LFA and $\mathrm{LFA}+\mathrm{TA}\textsc{cq}$ need to learn benchmark problems.

Protocol

- We generate an ordered sequence of examples,
- Both methods learn from the same sequence,
- We record the number of examples needed to reach 100% accuracy.

Experimental Evaluation

Problem	**Examples for 100% accuracy**		**Reduction**
	LFA	LFA+TACQ	
Sudoku	120	**80**	33%
Jigsaw [*3 instances*]	497	**377**	24%
Nurse Rostering [*3 instances*]	240	**197**	18%
Exam Timetabling [*3 instances*]	845	**306**	64%
Schur's Lemma	560	560	0%
Subgraph Isomorphism	640	640	0%
Golomb Ruler (10 variables)	2100	2100	0%
8-Queens	-	-	-

Experimental Evaluation

Problem	**Examples for 100% accuracy**		**Reduction**
	LFA	LFA+TACQ	
Sudoku	120	**80**	33%
Jigsaw [*3 instances*]	497	**377**	24%
Nurse Rostering [*3 instances*]	240	**197**	18%
Exam Timetabling [*3 instances*]	845	**306**	64%
Schur's Lemma	560	560	0%
Subgraph Isomorphism	640	640	0%
Golomb Ruler (10 variables)	2100	2100	0%
8-Queens	-	-	-

Experimental Evaluation

Problem	**Examples for 100% accuracy**		**Reduction**
	LFA	LFA+TACQ	
Sudoku	120	**80**	33%
Jigsaw [*3 instances*]	497	**377**	24%
Nurse Rostering [*3 instances*]	240	**197**	18%
Exam Timetabling [*3 instances*]	845	**306**	64%
Schur's Lemma	560	560	0%
Subgraph Isomorphism	640	640	0%
Golomb Ruler (10 variables)	2100	2100	0%
8-Queens	-	-	-

Experimental Evaluation

Problem	**Examples for 100% accuracy**		**Reduction**
	LFA	LFA+TAcq	
Sudoku	120	**80**	33%
Jigsaw [*3 instances*]	497	**377**	24%
Nurse Rostering [*3 instances*]	240	**197**	18%
Exam Timetabling [*3 instances*]	845	**306**	64%
Schur's Lemma	560	560	0%
Subgraph Isomorphism	640	640	0%
Golomb Ruler (10 variables)	2100	2100	0%
8-Queens	-	-	-

Experimental Evaluation

Problem	**Examples for 100% accuracy**		**Reduction**
	LFA	LFA+TAcq	
Sudoku	120	**80**	33%
Jigsaw [*3 instances*]	497	**377**	24%
Nurse Rostering [*3 instances*]	240	**197**	18%
Exam Timetabling [*3 instances*]	845	**306**	64%
Schur's Lemma	560	560	0%
Subgraph Isomorphism	640	640	0%
Golomb Ruler (10 variables)	2100	2100	0%
8-Queens	-	-	-

Results: Learning Interpretable Attributes

TACQ learns attributes corresponding to meaningful features

Results: Learning Interpretable Attributes

TACQ learns attributes corresponding to meaningful features

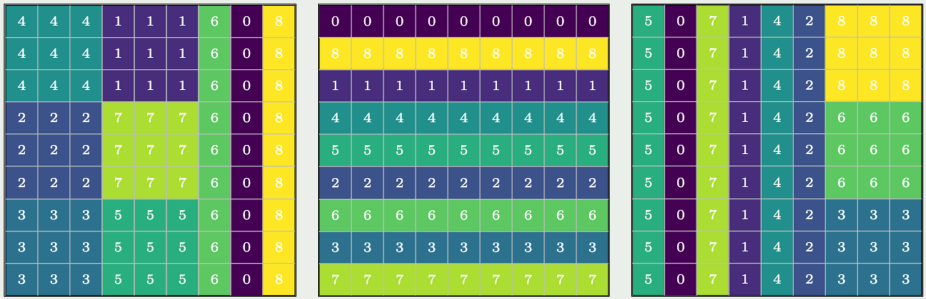


Figure: Illustration of the three attributes learned by TACQ for Sudoku.

1 | Refined simplicity of relations

Path: Define simplicity with a construction cost.

- Define a basis of primitives (e.g., $\{=, <, >, \ldots\}$);
- Favor relations constructible from these primitives with few operations.

2 | Better interpretability of attributes

Path: Bias towards interpretable attributes.

- Leverage statistical analysis of existing CP models (e.g., CSPLib);
- Guide learning to prefer this attribute distribution.

3 | Robustness to noise

Limitation: LFA/TACQ treat examples as ground truth.

- In some scenarios labels may be incorrect;
- **Path**: Relax the Max-SAT model.

Minimize (*Model Complexity* + *Examples Classification Error*)

4 | New class of oracles

Challenge: Use **black-box models** (e.g., CNNs) as oracles instead of humans.

Path: Leverage explanation methods (e.g., Grad-CAM for CNNs) to guide acquisition.

Conclusion

Summary

Problem: Reliance on prior knowledge prevents automated modeling.

- LFA: Enables learning without a language.
- TAcq: Recovers structure for better generalization.

Conclusion

Summary

Problem: Reliance on prior knowledge prevents automated modeling.

- LFA: Enables learning without a language.
- TAcq: Recovers structure for better generalization.

LFA and TAcq are publicly available:

LFA

Published at IJCAI-2023

Source code: language-free-acq

```
pip install languageFreeAcq
```

TAcq

Published at ECAI-2025

Source code: TAcq

```
pip install tacq
```

Thank you for your time and attention.

The work presented during this defense was supported by the TAILOR project, funded by the EU Horizon 2020 research and innovation programme under GA No 952215, by the AI Interdisciplinary Institute ANITI, funded by the French program "Investing for the Future – PIA3" under GA No ANR-19-PI3A-0004, and by the ANR AXIAUM project ANR-20-THIA-0005-01 (Data Science Institute of the University of Montpellier).

Experiments were performed with the support of the ISDM-MESO platform at the University of Montpellier.

References

[1] Nicolas Beldiceanu and Helmut Simonis. "A Model Seeker: Extracting Global Constraint Models from Positive Examples". In: *Principles and Practice of Constraint Programming - 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*. Ed. by Michela Milano. Vol. 7514. Lecture Notes in Computer Science. Springer, 2012, pp. 141–157. DOI: 10.1007/978-3-642-33558-7_13. URL: https://doi.org/10.1007/978-3-642-33558-7_13.

[2] Christian Bessiere et al. "Acquiring Constraint Networks Using a SAT-based Version Space Algorithm". In: *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*. AAAI Press, 2006, pp. 1565–1568. URL: http://www.aaai.org/Library/AAAI/2006/aaai06-251.php.

[3] Christian Bessiere et al. "Constraint acquisition". In: *Artif. Intell.* 244 (2017), pp. 315–342. DOI: 10.1016/j.artint.2015.08.001. URL: https://doi.org/10.1016/j.artint.2015.08.001.

[4] Mohit Kumar, Samuel Kolb, and Tias Guns. "Learning Constraint Programming Models from Data Using Generate-And-Aggregate". In: *28th International Conference on Principles and Practice of Constraint Programming, CP 2022, July 31 to August 8, 2022, Haifa, Israel*. Ed. by Christine Solnon. Vol. 235. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 29:1–29:16. DOI: 10.4230/LIPIcs.CP.2022.29. URL: https://doi.org/10.4230/LIPIcs.CP.2022.29.

[5] Steven D. Prestwich et al. "Classifier-based constraint acquisition". In: *Ann. Math. Artif. Intell.* 89.7 (2021), pp. 655–674. DOI: 10.1007/s10472-021-09736-4. URL: https://doi.org/10.1007/s10472-021-09736-4.