Constraint Acquisition Based on Solution Counting

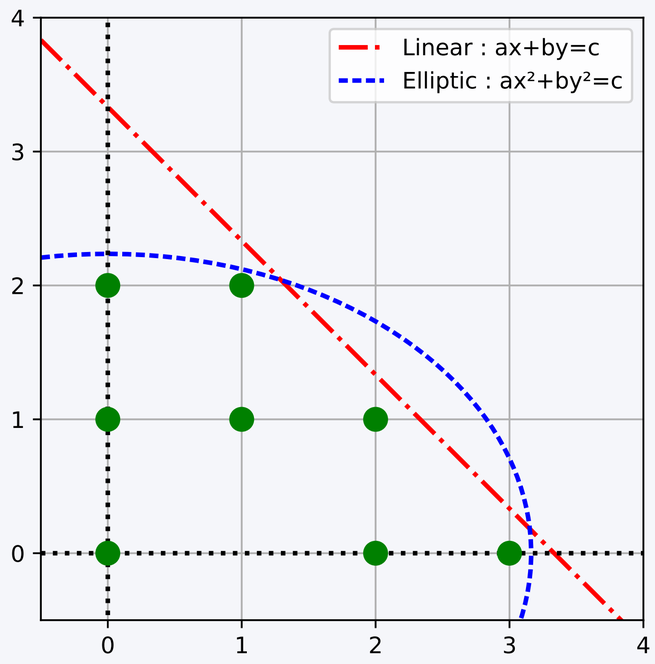
Christopher Coulombe [1] Claude-Guy Quimper [1]

[1]Université Laval, Québec, Canada

Abstract

We propose **CABSC**, a system that performs Constraint Acquisition Based on Solution Counting. In order to learn a Constraint Satisfaction Problem (CSP), the user provides positive examples and a **Meta-CSP**, i.e. a model of a combinatorial problem whose solution is a CSP. It allows listing the potential constraints that can be part of the CSP the user wants to learn. It also allows stating the parameters of the constraints and imposing constraints over these parameters. The CABSC reads the Meta-CSP and **returns the CSP that accepts the fewest solutions among the CSPs accepting all positive examples**. This is done using a branch and bound where the bounding mechanism makes use of a model counter. Experiments show that CABSC is successful at learning constraints and their parameters from positive examples.

Motivation & Simple Example



- The 8 points are positive examples of solutions
- We want to learn a CSP that explains the points
- We learn either a linear equation or an ellipse
- We know the points are in the first quadrant
- We need to find a, b, and c that define a CSP accepting all sample solutions while minimizing the number of feasible solutions.
- Here, the solution is an ellipse with $a = 1$, $b = 2$ and $c = 10$.

Gist of the idea

Our approach

CABSC learns a CSP by creating a **Constraint Optimization Problem** whose solution is the CSP that we aim to learn.

We search for the CSP that accepts all example solutions, but whose solution space is the smallest.

Meta-CSP

Definition - Meta-CSP

A **Meta-CSP** is a combinatorial optimization problem whose solution is itself a constraint optimization problem (CSP).

Features

A Meta-CSP model has :

1. a mapping from the decision variables to the solution examples
2. the parameters of the constraints as new decision variables
3. the known constraints that partially explain the example solutions
4. constraints that potentially explain the example solutions

Meta-CSP - Activation variables

Definition - Activation variable

An **activation variable** is a Boolean decision variable that is associated to a constraint. The constraint associated to the variable is included in the learned model if and only if the variable is set to **True**. If the variable is set to **False**, the solutions of the model can still satisfy the constraint.

Additional features

1. In a Meta-CSP model, constraints can be applied over activation variables and/or the constraints parameters.
2. The Meta-CSP model allows defining a family of CSPs from which we want to find the CSP that explains best the sample solutions.

Meta-CSP - Example

Following the examples with the 8 points.

```
set: domain = 1..10;

% Points are (x,y)
array: x_y=[1..2];
array: x=[1];
array: y=[2];

% Constraint parameters (considered as variables)
var domain: a;
var domain: b;
var domain: c;

% Activation variables
var 0..1: active1;
var 0..1: active2;

% Known constraints
    % x >= 0
constraint Linear(x, [1], ">=", 0, true);
    % y >= 0
constraint Linear(y, [1], ">=", 0, true);

% Potential constraints
    % a*x + b*y <= c
constraint Linear(x_y, [a,b], "<=", c, active1);
    % a*x^2 + b*y^2 <= c
constraint Ellipse(x_y, [a,b], "<=", c, active2);

% Constraints over activation variables
constraint Xor(active1, active2, true);
```

Model counting

Definition - Model counter

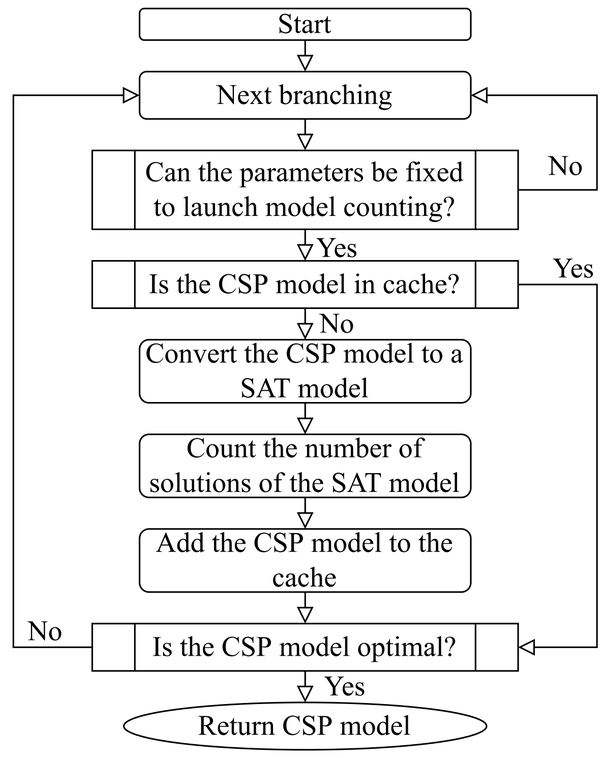
A **model counter** is a tool that counts, but not necessarily enumerates, the number of solutions of a CSP. A model counter is **exact** if it gives an exact count, **probabilistic exact** if it gives an exact count with configurable probability, and **approximate** if it gives a configurable approximation of the number of solutions.

We used `GANAK` [Sharma et al. IJCAI-19] as a **probabilistic exact model counter** and considered the return values *exact*. We also used `ApproxMC` [Chakraborty et al. IJCAI-16, Soos et al. AAAI-19] as an **approximate model counter** to quickly remove CSPs from the search tree.

How CABSC works

1. A Meta-CSP model that lists the known and potential constraints is created
2. Branch and bound is used where the nodes represent partial CSPs
3. A leaf in the search tree is a CSP
4. The bounding mechanism rewrites the CSPs as SAT models. The model counters then count the number of solutions of the CSPs.
5. CABSC uses a simple cache to avoid redundant queries to the model counters

Flowchart



Experimentations

Goal

We created **4 benchmarks** of instances that represent a nurse rostering problem. The goal was to learn the model that generated the sample schedules.

Benchmarks

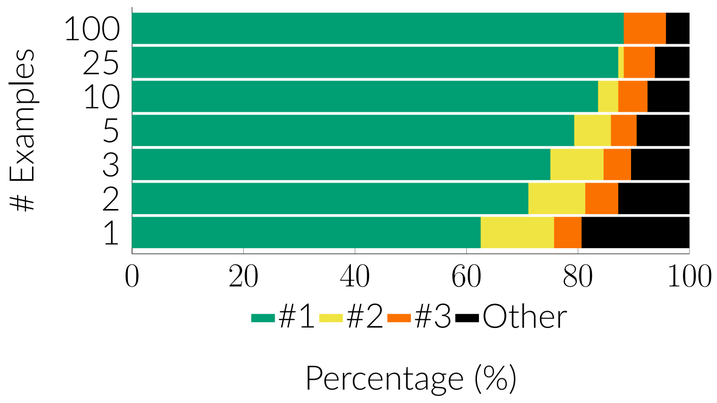
The benchmarks were created using conjunctions of the **Sequence constraint** or the **Among constraint** [Beldiceanu et al. Math. and Comp. Modelling]. The benchmarks included simplified instances, instances with known constraints (a partial model), instances with **vacations** and instances with **overtime**.

Meta-CSP

The Meta-CSP was written so that the solver had to select which constraint to learn among Sequence and Among. It also had to learn the correct parameters.

Results

The **3 CSPs** with the smallest solution space are learned. The following graph shows for one of the benchmark whether the CSP used to generate the examples is among the 3 first CSPs learned by CABSC. If the correct CSP was not learned, the instance was sorted with "Other".



Conclusion et contributions

- **CABSC** is a new way to do constraint acquisition
- Our approach learns the CSP that **minimizes the size of its solution space**s
- The approach do **not assume independence** between decision variables
- CABSC learns **multiple constraints** and takes into account **partial models**
- CABSC successfully learns CSPs and requires **few examples**

Code: http://www2.ift.ulaval.ca/quimper/publications.php

28th International Conference on Principles and Practice of Constraint Programming (CP 2022), Haifa

christopher.coulombe.1@ulaval.ca claude-guy.quimper@ift.ulaval.ca