Learning Efficient Constraints in Answer Set Programming

Alice Tarzariol, Martin Gebser, Konstantin Schekotihin

Alpen-Adria-Universität Klagenfurt, Austria

March 8, 2024

Overview

1. Answer Set Programming
2. Practical Issues with Symmetries
3. Learning Efficient Constrains
4. Extension for Optimization Problems
5. Conclusions and Future Works

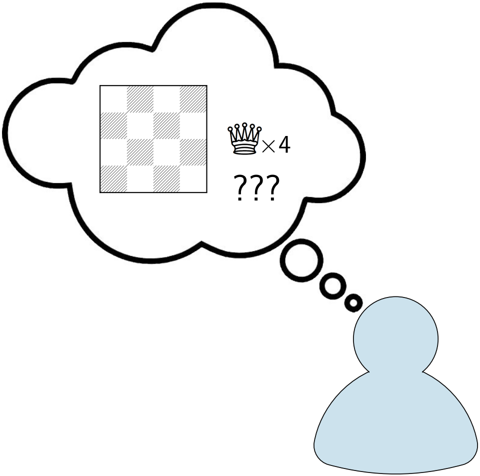
Answer Set Programming

Introduction

- Declarative programming paradigm, based on logic
- Suited for combinatorial search and optimization problems
- For examples:
  - $\rightarrow$ Configurations
  - $\rightarrow$ Scheduling
  - $\rightarrow$ Planning
  - $\rightarrow$ $\dots$
- Strengths: flexible, intuitive, expressive, efficient systems

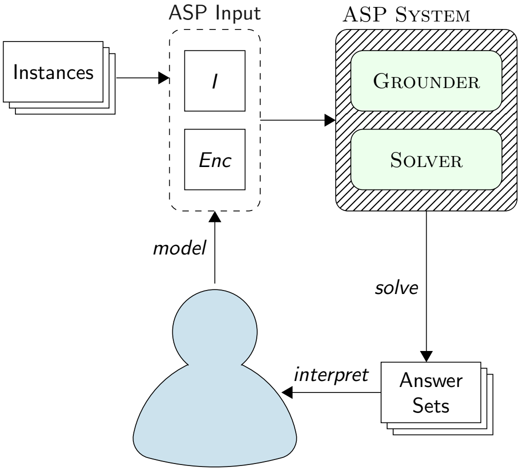
ASP Declarative Approach

"Describe the problem/solutions" VS "List the steps to solve it"



ASP Declarative Approach

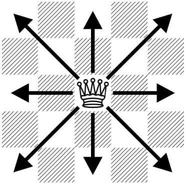
"Describe the problem/solutions" VS "List the steps to solve it"



ASP Input

When the users model a problem, they define two elements:

- **Encoding:**
  - → General rules
  - → Describe objects and relations in the problem
  - → E.g., grid, queens attack

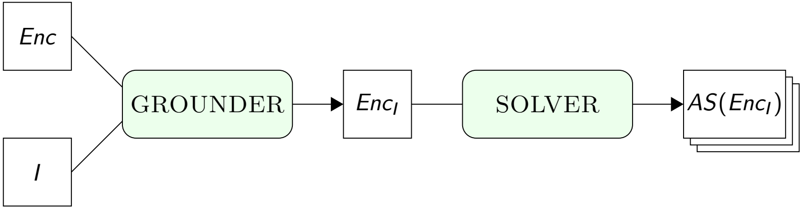


$\times n$

- **Instance:**
  - → Specific input
  - → Drawn from a set of valid instances
  - → Number(s), graph, etc.

$$n = 4$$

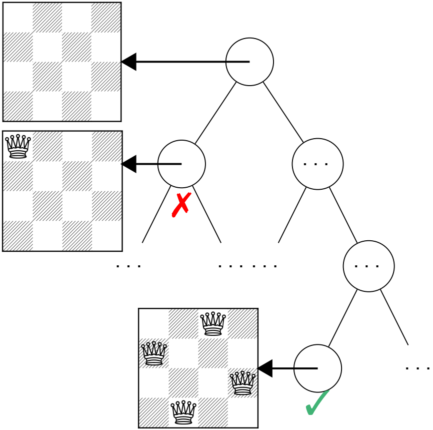
ASP System



1. GROUNDER:
   - → Combines the information between *Enc* and *I*
   - → Result: Enc_I, which defines concrete elements and conditions

2. SOLVER
   - → Searches for the truth assignments that are answer sets (solutions)

(Intuitive) Solving Procedure



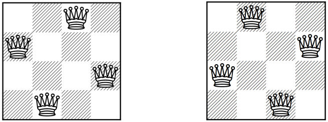
Practical Issues with Symmetries

Issues with Industrial Applications

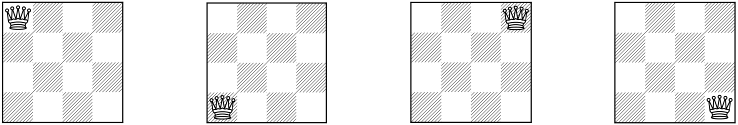
- Huge instances, no answer
- Several techniques for coping with different issues
- In this talk: symmetries!
- The encoding *Enc* highly influences the search performance
- It should avoid symmetries, exploit invariants of instances, etc.

What is a Symmetry?

- A symmetry entails equivalent characteristics to another object, e.g., truth assignments
- Symmetric solutions



- Symmetric (partial) interpretations



How do we deal with symmetries?

- Modelling a problem to avoid symmetric solutions is hard
- Automatically identify symmetric solutions and extend a given model with constraints discarding them

How do we deal with symmetries?

- Modelling a problem to avoid symmetric solutions is hard
- Automatically identify symmetric solutions and extend a given model with constraints discarding them
  - → Instance-specific
  - → Model-oriented

Instance-Specific Approaches to Symmetry Breaking

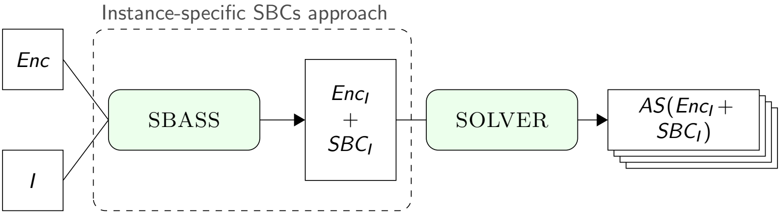
Remove the symmetries according to syntactic properties of the current problem instance [4] by adding *Symmetry Breaking Constraints (SBCs)*

Instance-Specific SBCs

1. Define a reduction to graph automorphism problem
2. Use tools to find symmetric vertex permutations, e.g. SAUCY [2]
3. Derive symmetries of the original problem from vertex permutations
4. Add SBCs according to a certain criterion (e.g., lexicographic order)

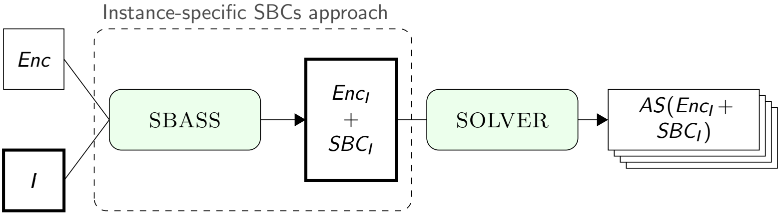
Symmetry Breaking Answer Set Solving

SBASS [3] implements automatic symmetry detection and breaking for ground ASP programs



Symmetry Breaking Answer Set Solving

SBASS [3] implements automatic symmetry detection and breaking for ground ASP programs

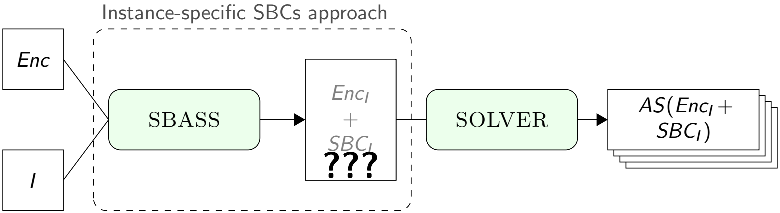


Limitations of instance-specific SBCs approaches

1. No generalisation

Symmetry Breaking Answer Set Solving

SBASS [3] implements automatic symmetry detection and breaking for ground ASP programs

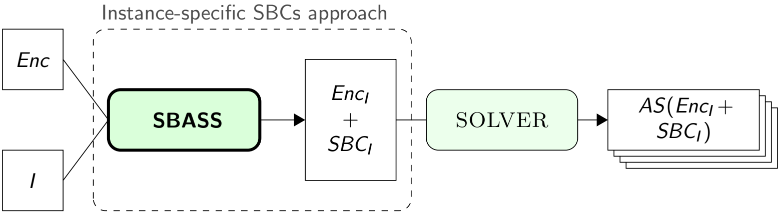


Limitations of instance-specific SBCs approaches

1. No generalisation
2. Interpretability

Symmetry Breaking Answer Set Solving

SBASS [3] implements automatic symmetry detection and breaking for ground ASP programs

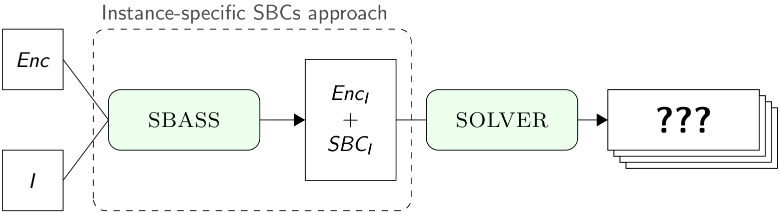


Limitations of instance-specific SBCs approaches

1. No generalisation
2. Interpretability
3. Pre-processing overhead

Symmetry Breaking Answer Set Solving

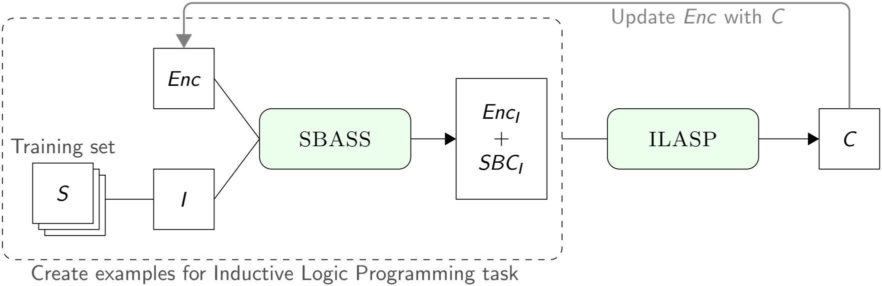
SBASS [3] implements automatic symmetry detection and breaking for ground ASP programs



Limitations of instance-specific SBCs approaches

1. No generalisation
2. Interpretability
3. Pre-processing overhead
4. Redundancy

Model-Oriented Approach



Learning Framework

Input: "naive" encoding Enc, set of representative satisfiable instances S

$\rightarrow$ Identify symmetries of each instance in S

$\rightarrow$ Use the symmetries to create examples for an ILP task

Output: constraints C pruning redundant parts of the search space

Learning Efficient Constrains

Learning Efficient Constraints in ASP

Version 1	Version 2	Version 3
Simple Decision Problems	Complex Decision Problems	Optimization Problems
IJCAI 2021 [5] & MLJ 2022 [6]	ICLP 2022 [8]	AAAI 2023 [7]

Research Goals

1. Define a model-oriented approach capable of lifting symmetries and obtaining first-order constraints (for a target distribution)
2. Investigate extensions to enable learning constraints for advanced combinatorial problems
3. Extend the expressiveness of the learning framework to analyse the symmetries of optimization problems

Learning Efficient Constraints in ASP

Version 1	Version 2	Version 3
Simple Decision Problems	**Complex Decision Problems**	Optimization Problems
IJCAI 2021 [5] & MLJ 2022 [6]	ICLP 2022 [8]	AAAI 2023 [7]

Research Goals

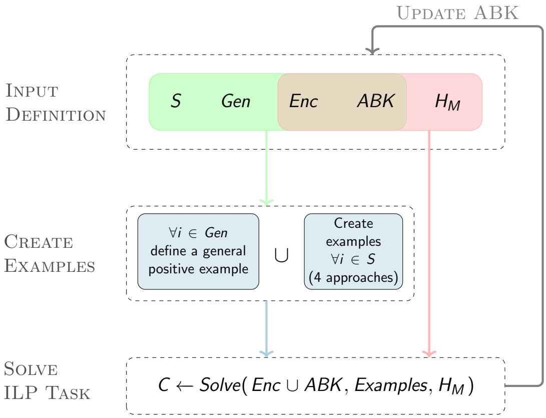
1. Define a model-oriented approach capable of lifting symmetries and obtaining first-order constraints (for a target distribution)
2. Investigate extensions to enable learning constraints for advanced combinatorial problems
3. Extend the expressiveness of the learning framework to analyse the symmetries of optimization problems

Learning Framework

Our learning framework is based on Inductive Logic Programming (ILP), a form of learning that extends given background knowledge with new hypotheses explaining a set of positive and negative examples

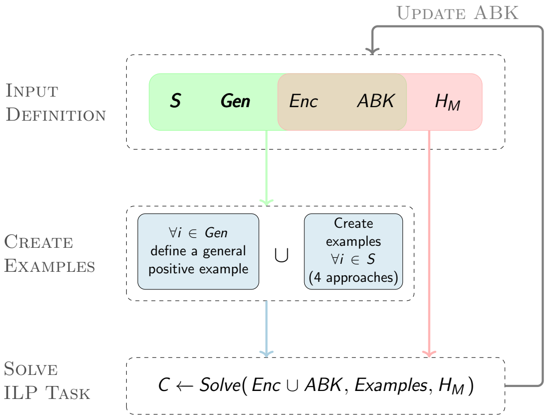
Learning Framework

Our learning framework is based on Inductive Logic Programming (ILP), a form of learning that extends given background knowledge with new hypotheses explaining a set of positive and negative examples



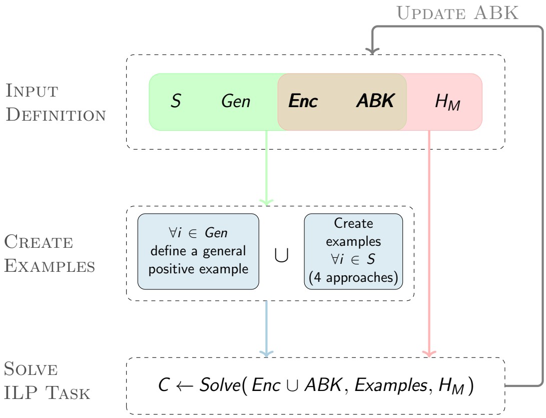
Learning Framework

Our learning framework is based on Inductive Logic Programming (ILP), a form of learning that extends given background knowledge with new hypotheses explaining a set of positive and negative examples



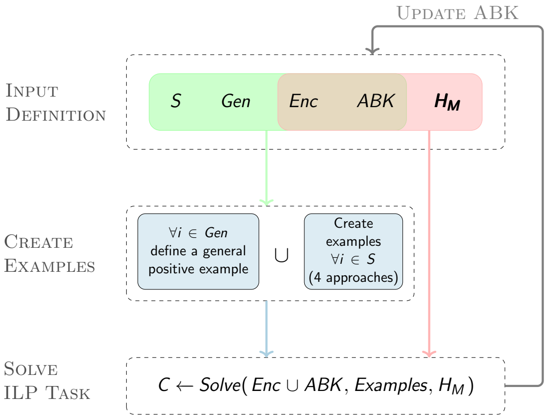
Learning Framework

Our learning framework is based on Inductive Logic Programming (ILP), a form of learning that extends given background knowledge with new hypotheses explaining a set of positive and negative examples

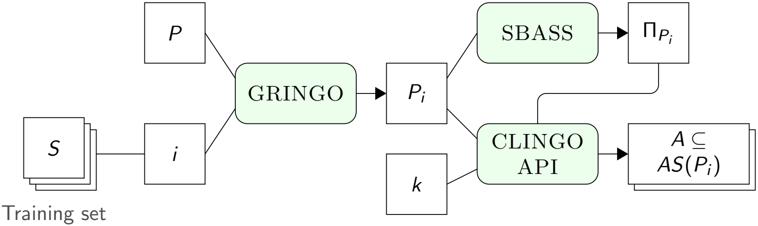


Learning Framework

Our learning framework is based on Inductive Logic Programming (ILP), a form of learning that extends given background knowledge with new hypotheses explaining a set of positive and negative examples



Scalable Approach for Creating Examples



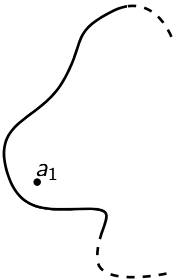
Scalable Full-SBCs

- Exploit CLINGO API with the symmetries Π_{P_i} to explore only k cells from the partition of $AS(P_i)$
- For each cell, the smallest element is a positive example, then create MAX negative examples

Scalable Approach for Creating Examples (Cont.)

For each instance $i \in S$ (set of representative satisfiable instances in input):

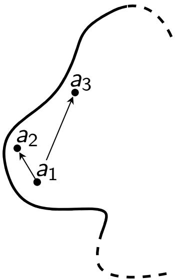
1. SBASS(P_i) returns Π_{P_i}
2. CLINGO returns $a_1 \in AS(P_i)$



Scalable Approach for Creating Examples (Cont.)

For each instance $i \in S$ (set of representative satisfiable instances in input):

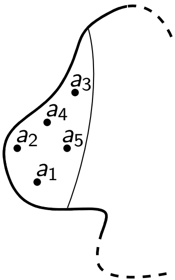
1. $\textsc{Sbass}(P_i)$ returns Π_{P_i}
2. $\textsc{Clingo}$ returns $a_1 \in AS(P_i)$
3. Apply Π_{P_i} to a_1 to identify its symmetries



Scalable Approach for Creating Examples (Cont.)

For each instance $i \in S$ (set of representative satisfiable instances in input):

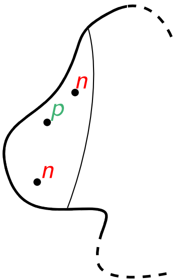
1. $\textsc{Sbass}(P_i)$ returns Π_{P_i}
2. $\textsc{Clingo}$ returns $a_1 \in AS(P_i)$
3. Apply Π_{P_i} to a_1 to identify its symmetries



Scalable Approach for Creating Examples (Cont.)

For each instance $i \in S$ (set of representative satisfiable instances in input):

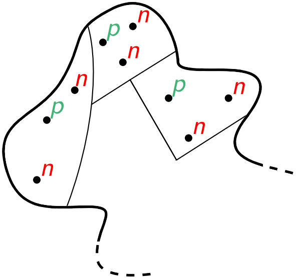
1. $\textsc{Sbass}(P_i)$ returns Π_{P_i}
2. $\textsc{Clingo}$ returns $a_1 \in AS(P_i)$
3. Apply Π_{P_i} to a_1 to identify its symmetries
4. One positive example, MAX sampled negative examples



Scalable Approach for Creating Examples (Cont.)

For each instance $i \in S$ (set of representative satisfiable instances in input):

1. $\textsc{Sbass}(P_i)$ returns Π_{P_i}
2. $\textsc{Clingo}$ returns $a_1 \in AS(P_i)$
3. Apply Π_{P_i} to a_1 to identify its symmetries
4. One positive example, MAX sampled negative examples
5. Repeat the previous steps k times or until no new solutions are found



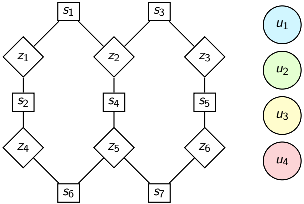
Partner Units Problem (PUP)

- Abstract representation of a configuration problem originating from a railway safety system of Siemens
- Extremely hard to solve this problem with real, large-scale instances
- “Naive” encoding + “standard” instances $\rightarrow$ 99% symmetric candidate solutions

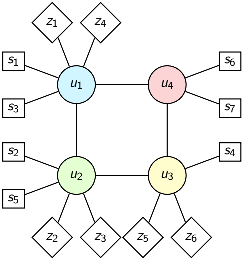
Simple ASP Encoding for PUP [1]

```
% Input
zone(Z) :- zone2sensor(Z,D).
sensor(D) :- zone2sensor(Z,D).
comUnit(1..n).

% Generate
1 { unit2zone(U,Z) : comUnit(U) } 1 :- zone(Z).
1 { unit2sensor(U,S) : comUnit(U) } 1 :- sensor(S).

% Constraint UCAP
:- comUnit(U), ucap+1 { unit2zone(U,Z): zone(Z) }.
:- comUnit(U), ucap+1 { unit2sensor(U,S): sensor(S) }.

% Constraint IUCAP
partnerunits(U,P) :- unit2zone(U,Z), zone2sensor(Z,S),  unit2sensor(P,S), U!=P.
partnerunits(U,P) :- partnerunits(P,U), comUnit(U), comUnit(P).
:- comUnit(U), iucap+1 { partnerunits(U,P): comUnit(P) }.
```

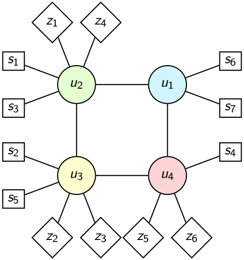
PUP Example



(a) Partner Units Problem instance with $UCAP = IUCAP = 2$

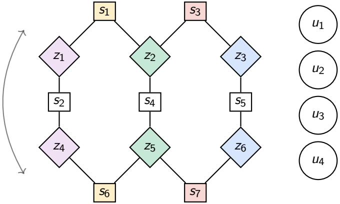


(b) A solution for the instance

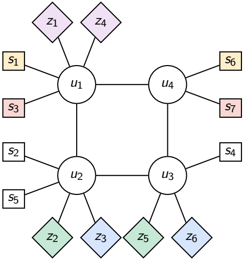


(c) A symmetric solution

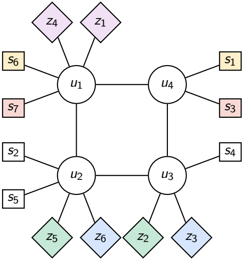
PUP Example



(a) Partner Units Problem instance with
$UCAP = IUCAP = 2$



(b) A solution for the instance



(c) A symmetric solution

Experiments

	Enc	SBASS	Enc+SBCs(i)	Enc+ABK	Adv
dbl-10	0.02	0.04	0.02	**0.01**	**0.01**
un-dbl-10*	505.91	0.03	TO	**0.01**	0.16
dbl-20	1.06	0.31	1.40	**0.08**	0.53
un-dbl-20*	TO	0.28	TO	**0.34**	TO
dbl-30	1.70	3.12	0.91	**0.46**	2.21
un-dbl-30*	TO	3.19	TO	**19.97**	TO
dbl-40	TO	14.58	482.17	**5.50**	11.50
un-dbl-40*	TO	9.52	TO	**65.54**	TO
dbl-50	TO	57.91	TO	**54.89**	542.09

- Three distributions of PUP instances: **double**, doublev, triple.
- **Enc+ABK**: the most efficient constraints obtained from the samples.
- **Adv**: advanced encoding containing static symmetric breaking and ordering rules.

Experiments

	Enc	SBASS	Enc+SBCs(i)	Enc+ABK	Adv
dbl-10	0.02	0.04	0.02	**0.01**	**0.01**
un-dbl-10*	505.91	0.03	TO	**0.01**	0.16
dbl-20	1.06	0.31	1.40	**0.08**	0.53
un-dbl-20*	TO	0.28	TO	**0.34**	TO
dbl-30	1.70	3.12	0.91	**0.46**	2.21
un-dbl-30*	TO	3.19	TO	**19.97**	TO
dbl-40	TO	14.58	482.17	**5.50**	11.50
un-dbl-40*	TO	9.52	TO	**65.54**	TO
dbl-50	TO	57.91	TO	**54.89**	542.09

- Three distributions of PUP instances: **double**, doublev, triple.
- **Enc+ABK**: the most efficient constraints obtained from the samples.
- **Adv**: advanced encoding containing static symmetric breaking and ordering rules.

Experiments

	Enc	SBASS	Enc+SBCs(i)	Enc+ABK	Adv
dbl-10	0.02	0.04	0.02	**0.01**	**0.01**
un-dbl-10*	505.91	0.03	TO	**0.01**	0.16
dbl-20	1.06	0.31	1.40	**0.08**	0.53
un-dbl-20*	TO	0.28	TO	**0.34**	TO
dbl-30	1.70	3.12	0.91	**0.46**	2.21
un-dbl-30*	TO	3.19	TO	**19.97**	TO
dbl-40	TO	14.58	482.17	**5.50**	11.50
un-dbl-40*	TO	9.52	TO	**65.54**	TO
dbl-50	TO	57.91	TO	**54.89**	542.09

- Three distributions of PUP instances: **double**, doublev, triple.
- **Enc+ABK**: the most efficient constraints obtained from the samples.
- **Adv**: advanced encoding containing static symmetric breaking and ordering rules.

Learning Efficient Constraints in ASP

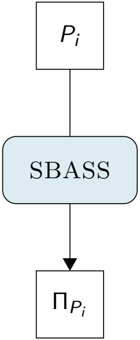
Version 1	Version 2	Version 3
Simple Decision Problems	Complex Decision Problems	**Optimization Problems**
IJCAI 2021 [5] & MLJ 2022 [6]	ICLP 2022 [8]	AAAI 2023 [7]

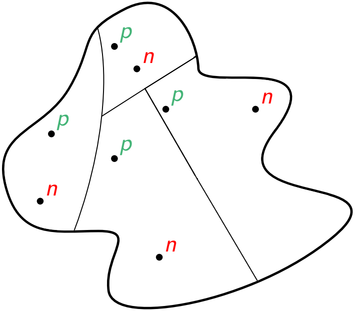
Research Goals

1. Define a model-oriented approach capable of lifting symmetries and obtaining first-order constraints (for a target distribution)
2. Investigate extensions to enable learning constraints for advanced combinatorial problems
3. **Extend the expressiveness of the learning framework to analyse the symmetries of optimization problems**

Extension for Optimization Problems

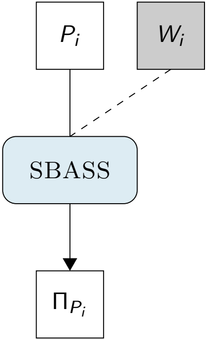
Decision vs Optimization Problems

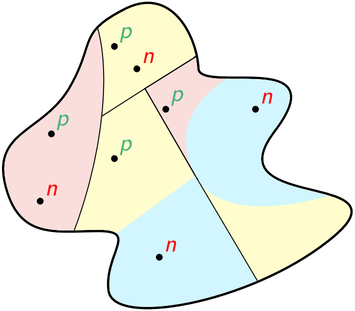




- For decision problems, correct partition and representative solutions

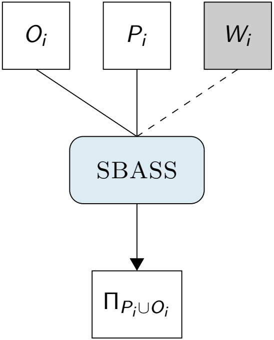
Decision vs Optimization Problems

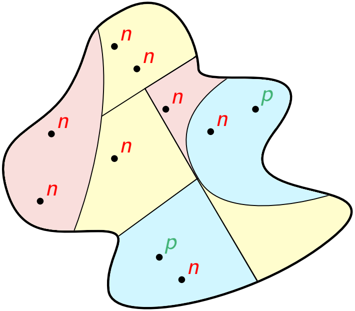




- For decision problems, correct partition and representative solutions
- SBASS doesn't parse weak constraints, thus incorrect examples

Decision vs Optimization Problems





- For decision problems, correct partition and representative solutions
- SBASS doesn't parse weak constraints, thus incorrect examples
- *Solution*: add auxiliary normal rules defining a finer partition

Conclusions and Future Works

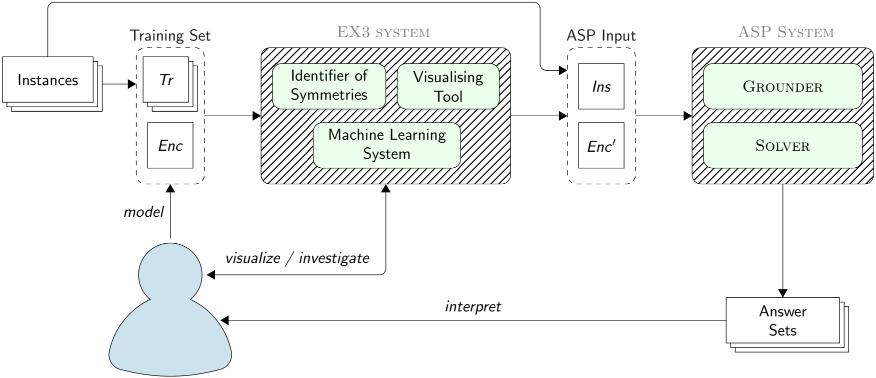
Conclusions

- Design and implement a model-oriented approach for lifting SBCs
- Test it over some simple and advanced combinatorial problems (decision and optimization)
- The learned constraints:
  - $\rightarrow$ are general and easier to interpret than ground SBCs
  - $\rightarrow$ obtain better-solving performance than the original program, the online application of instance-specific SBCs approaches, and SBCs designed by experts (for some type of instances)
- But wait ... there's more!

Current Limitations

- Instances labelling has a critical impact on the framework applicability
  - $\rightarrow$ Automatically extract features and generalise them
- No transferable knowledge/no insights from the problem or instance distribution
  - $\rightarrow$ Exploit the properties learned to define language bias (i.e., auxiliary predicates)
- Interpretability of the learned constraints
  - $\rightarrow$ Explain to the user the meaning of the constraints learned

EX3 – EXtract, EXploit and EXplain Knowledge



Bibliography I

[1] Markus Aschinger, Conrad Drescher, Gerhard Friedrich, Georg Gottlob, Peter Jeavons, Anna Ryabokon, and Evgenij Thorstensen.
Optimization methods for the partner units problem.
In *CPAIOR*, volume 6697 of *LNCS*, pages 4–19. Springer, 2011.

[2] P. Darga, H. Katebi, M. Liffiton, I. Markov, and K. Sakallah.
Saucy.
`http://vlsicad.eecs.umich.edu/BK/SAUCY/`, 2004.
Accessed: 2021-05-21.

[3] C. Drescher, O. Tifrea, and T. Walsh.
Symmetry-breaking answer set solving.
In M. Balduccini and S. Woltran, editors, *Proceedings of ICLP'10 Workshop on Answer Set Programming and Other Computing Paradigm*, 2010.

Bibliography II

[4] K. Sakallah.
Symmetry and satisfiability.
In A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 10, pages 289–338. IOS Press, 2009.

[5] A. Tarzariol, M. Gebser, and K. Schekotihin.
Lifting symmetry breaking constraints with inductive logic programming.
In Z. Zhi-Hua, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI'21)*, pages 2062–2068. ijcai.org, 2021.

[6] Alice Tarzariol, Martin Gebser, and Konstantin Schekotihin.
Lifting symmetry breaking constraints with inductive logic programming.
Machine Learning, 111(4):1303–1326, 2022.

[7] Alice Tarzariol, Martin Gebser, Konstantin Schekotihin, and Mark Law.
Learning to break symmetries for efficient optimization in answer set programming.
In *Proceedings of the Thirty-seventh AAAI Conference on Artificial Intelligence (AAAI'23)*. AAAI Press, 2023.

Bibliography III

[8] Alice Tarzariol, Konstantin Schekotihin, Martin Gebser, and Mark Law.
Efficient lifting of symmetry breaking constraints for complex combinatorial problems.
Theory and Practice of Logic Programming, 2022.