ApproxASP – A Scalable Approximate Answer Set Counter[a]

Mohimenul Kabir,

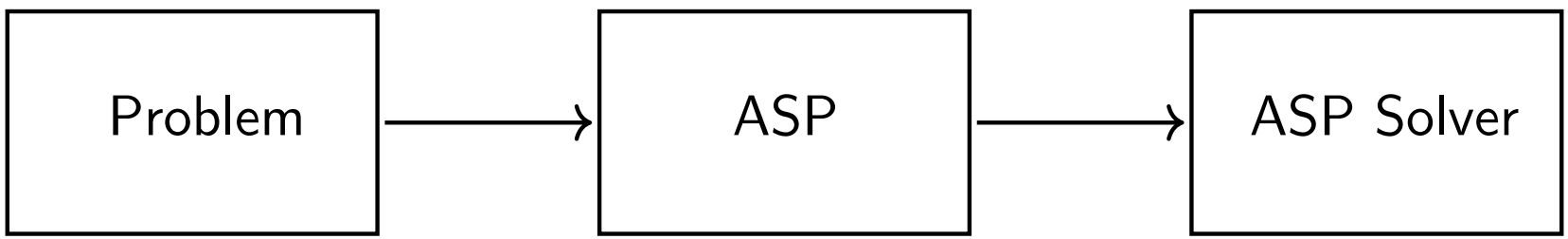
Joint work with: Flavio Everardo, Ankit Shukla, Johannes K. Fichte, Markus Hecher, Kuldeep S. Meel

National University of Singapore

[a] published in AAAI-2022

Answer Set Programming (ASP)

- ASP is a form of declarative programming:
  - logic programming
  - non-monotonic reasoning



Solution of problem $\equiv$ answer set of ASP

ASP Encoding

s-t Graph Reachability Problem

Problem Definition: Decides two nodes s and t are connected

ASP Encoding

s-*t* Graph Reachability Problem

Problem Definition: Decides two nodes s and t are connected

```
%Edges and nodes
edge(a,b).  edge(b,c).  ... node(a).  node(b).  ...
```

ASP Encoding

s-*t* Graph Reachability Problem

Problem Definition: Decides two nodes s and t are connected

```
%Edges and nodes
edge(a,b).  edge(b,c).  ... node(a).  node(b).  ...
% Start traversing from node s
reached(s).
```

ASP Encoding

s-*t* Graph Reachability Problem

Problem Definition: Decides two nodes *s* and *t* are connected

```
%Edges and nodes
edge(a,b).  edge(b,c).  ... node(a).  node(b).  ...
% Start traversing from node s
reached(s).
% Transitive definition of reachability
reached(Y) :- reached(X), edge(X,Y).
```

ASP Encoding

s-t Graph Reachability Problem

Problem Definition: Decides two nodes *s* and *t* are connected

```
%Edges and nodes
edge(a,b).  edge(b,c).  ... node(a).  node(b).  ...
% Start traversing from node s
reached(s).
% Transitive definition of reachability
reached(Y) :- reached(X), edge(X,Y).
% Node t must be reached
:- ~reached(t).
```

ASP: Syntax

- A *rule-based language* for problem encoding:

$$r : h_1 \vee \ldots \vee h_\ell \leftarrow b_1, \ldots, b_k, \sim b_{k+1}, \ldots, \sim b_{k+m}$$

$$Head(r) \leftarrow Body^+(r), \sim Body^-(r)$$

ASP: Syntax

- A *rule-based language* for problem encoding:

$$r : h_1 \vee \ldots \vee h_\ell \leftarrow b_1, \ldots, b_k, \sim b_{k+1}, \ldots, \sim b_{k+m}$$

$$Head(r) \leftarrow Body^+(r), \sim Body^-(r)$$

- A program $P \equiv$ *set of rules*.
- Program P is *disjunctive* if $\exists r \in P$, $\ell > 1$. Otherwise, program P is *normal*.

ASP: Syntax

- A *rule-based language* for problem encoding:

$$r : h_1 \vee \ldots \vee h_\ell \leftarrow b_1, \ldots, b_k, \sim b_{k+1}, \ldots, \sim b_{k+m}$$

$$Head(r) \leftarrow Body^+(r), \sim Body^-(r)$$

- A program $P \equiv$ *set of rules*.
- Program P is *disjunctive* if $\exists r \in P$, $\ell > 1$. Otherwise, program P is *normal*.
- The model of P is *answer set* or AS(P).
- Answer set solver enumerates AS(P) (ASP solving).

Answer Set Programming (ASP) (contd.)

GL-reduct

Given a program P and a set M of atoms

$$P^M = \{Head(r) \leftarrow Body^+(r) | r \in P, M \cap Body^-(r) = \emptyset\}$$

Answer Set

A set of atoms M is answer set of program P iff

- M is a model of P
- M is a (subset) minimal model of P^M

Motivation

Applications of ASP Counting

- Network Reliability (Aziz et al., 2015)
- Probabilistic Inference (Lee, Talsania, and Wang, 2017)
- Planning (Alrabbaa, Rudolph, and Schweizer, 2018)
- Navigation (Fichte, Gaggl, and Rusovac, 2022)

Challenges of Counting

- Exact answer set counting (Fichte et al., 2016)
  Limitation: Fails on higher treewidth (Hecher, 2020)

Challenges of Counting

- Exact answer set counting (Fichte et al., 2016)
  Limitation: Fails on higher treewidth (Hecher, 2020)
- Enumerate answer sets (Gebser, Kaufmann, and Schaub, 2012)
  Limitation: Fails to enumerate large number of answer sets

Challenges of Counting

- Exact answer set counting (Fichte et al., 2016)
  Limitation: Fails on higher treewidth (Hecher, 2020)
- Enumerate answer sets (Gebser, Kaufmann, and Schaub, 2012)
  Limitation: Fails to enumerate large number of answer sets
- ASP to SAT + run propositional model counter
  Limitation: ASP to SAT incurs exponential overhead

Challenges of Counting

- Exact answer set counting (Fichte et al., 2016)
  Limitation: Fails on higher treewidth (Hecher, 2020)
- Enumerate answer sets (Gebser, Kaufmann, and Schaub, 2012)
  Limitation: Fails to enumerate large number of answer sets
- ASP to SAT + run propositional model counter
  Limitation: ASP to SAT incurs exponential overhead

Solution: Approximate Answer Set Counting

Table of Content

Introduction

Preliminaries

Approximate Counting Technique

Implementation Details

Experiments

$(\epsilon, \delta)-$Approximate Counting

Input

An instance I, tolerance ϵ and confidence δ

Output

estimate cnt such that

$$\Pr\left[\frac{|\,\mathsf{AS}(I)|}{(1+\epsilon)} \leq \mathsf{cnt} \leq (1+\epsilon)\cdot|\,\mathsf{AS}(I)|\right] \geq 1-\delta.$$

Table of Content

Introduction

Preliminaries

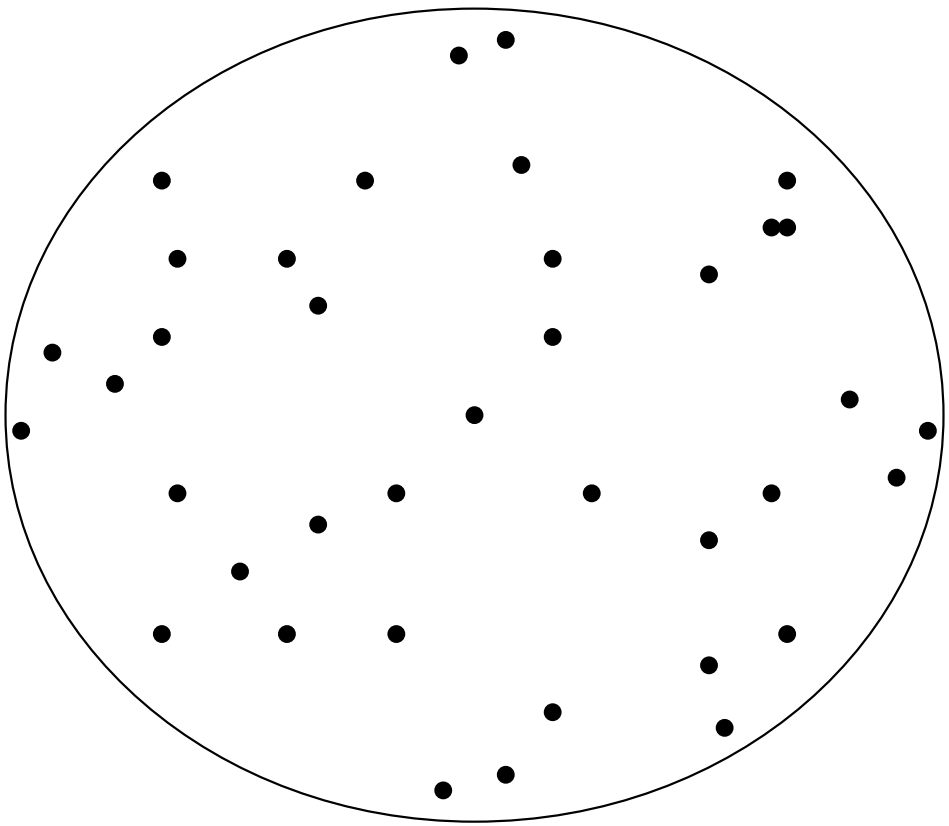
Approximate Counting Technique

Implementation Details

Experiments

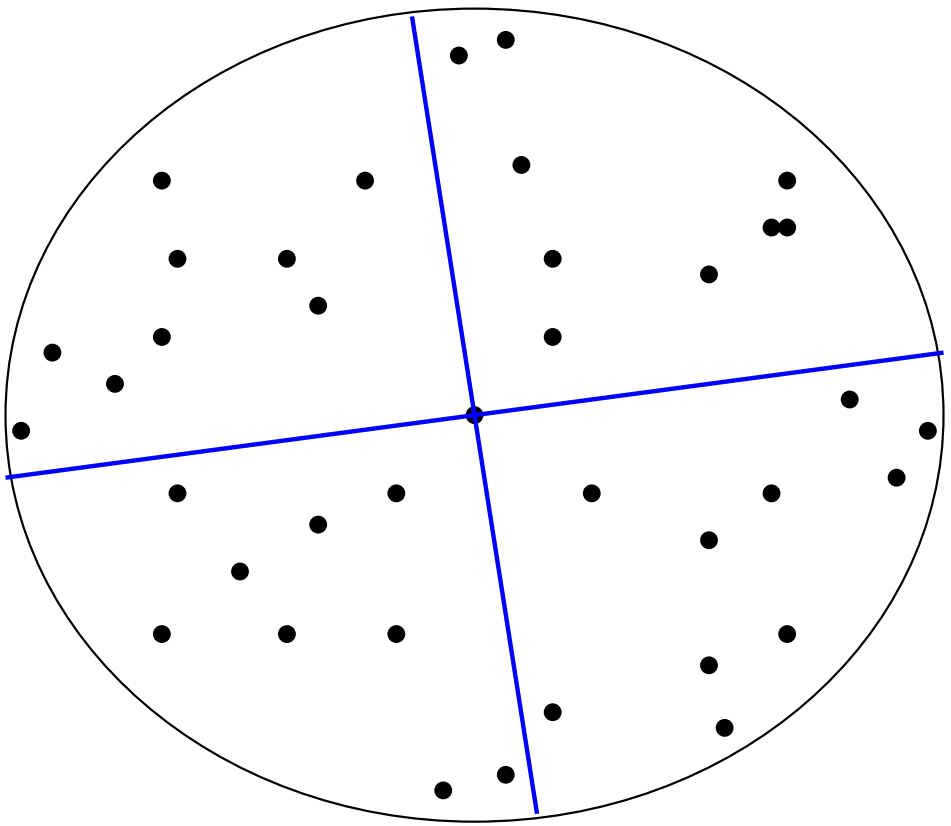
Approximate Counting

Basic Idea



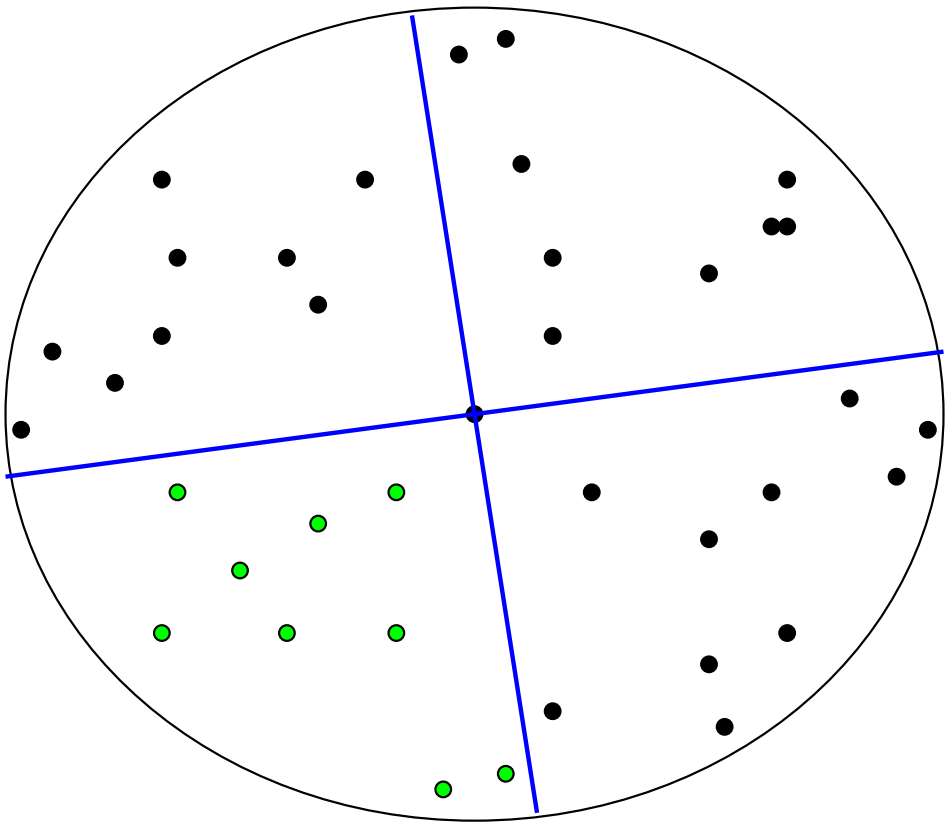
Approximate Counting

Basic Idea (Contd.)



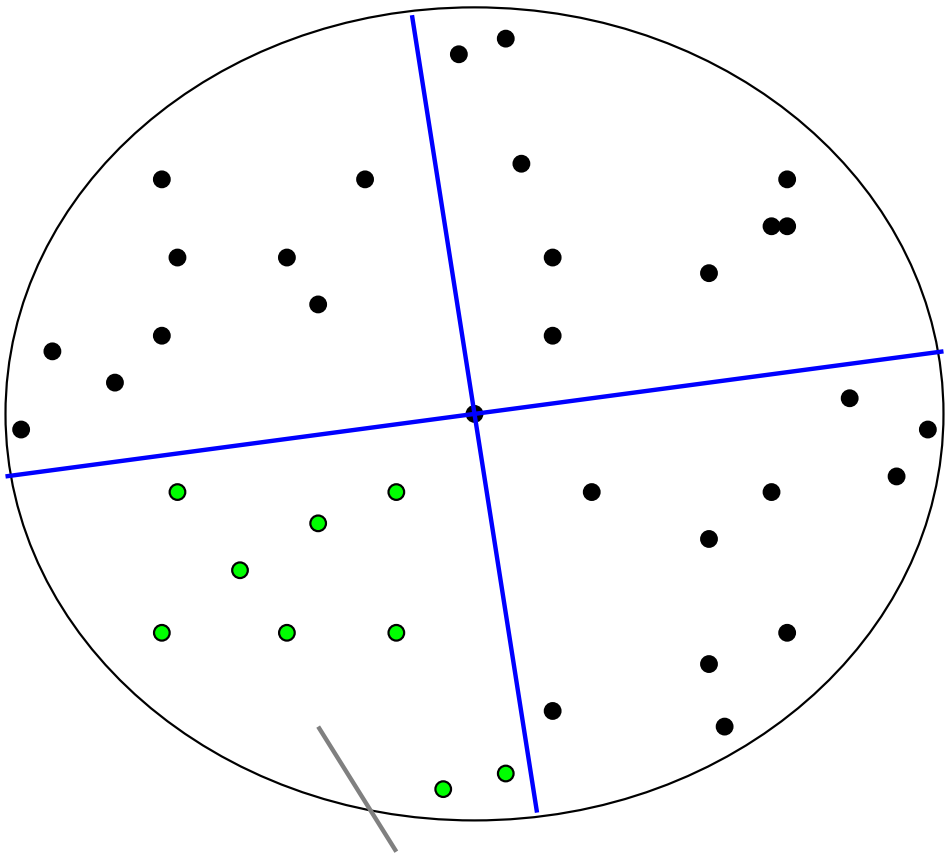
Approximate Counting

Basic Idea (Contd.)



Approximate Counting

Basic Idea (Contd.)



$$\boxed{\text{AppxCount} = \text{nModels} \times \text{nCells}}$$

Approximate Counting

High-level Overview

- Target: *Partition the search space uniformly and independently* (Stockmeyer, 1983) — *universal hash function* (Carter and Wegman, 1977) — Computationally Hard.

Approximate Counting

High-level Overview

- Target: *Partition the search space uniformly and independently* (Stockmeyer, 1983) — *universal hash function* (Carter and Wegman, 1977) — Computationally Hard.
- Partition Trick: Gomes et al. [GSS06] used 2-universal hash function for approximate counting.

Approximate Counting

High-level Overview

- Target: *Partition the search space uniformly and independently* (Stockmeyer, 1983) — *universal hash function* (Carter and Wegman, 1977) — Computationally Hard.
- Partition Trick: Gomes et al. [GSS06] used 2-universal hash function for approximate counting.
- 2-universal hash function: *random XOR constraints* where each variable belongs to each XOR with probability $\frac{1}{2}$.

Approximate Counting

High-level Overview

- Target: *Partition the search space uniformly and independently* (Stockmeyer, 1983) — *universal hash function* (Carter and Wegman, 1977) — Computationally Hard.
- Partition Trick: Gomes et al. [GSS06] used 2-universal hash function for approximate counting.
- 2-universal hash function: *random XOR constraints* where each variable belongs to each XOR with probability $\frac{1}{2}$.
- Implementation: ASP + XOR.

Table of Content

Introduction

Preliminaries

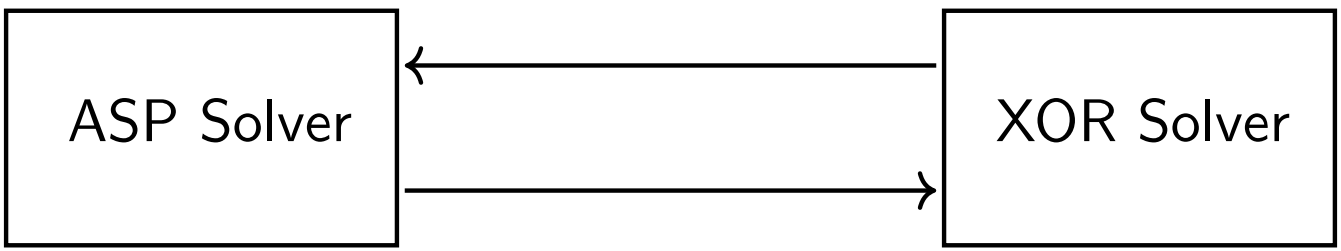
Approximate Counting Technique

Implementation Details

Experiments

Implementation Details

Architecture



- **ASP Solver**: Clingo (Gebser et al., 2007)
- **XOR Solver**: Han-Jiang's Gauss Jordan Elimination (GJE) (Han and Jiang, 2012)

Implementation Details

Optimizations in XOR Solving

- **Observation:** Longer XOR constraint makes the XOR solving harder (Soos, Gocht, and Meel, 2020)
- **Heuristic:** Constructing XOR constraints from *independent support* (IS)

Implementation Details

Optimizations in XOR Solving

- **Observation:** Longer XOR constraint makes the XOR solving harder (Soos, Gocht, and Meel, 2020)
- **Heuristic:** Constructing XOR constraints from *independent support* (IS)
- **Definition:** A set of atoms that uniquely defines the rest of atoms in every satisfying assignment.
- $P = \{a \vee \sim a.\ c \leftarrow \sim a.\ b \leftarrow a.\}$.
  $\{a\}$ is an independent support.

Implementation Details

IS for ASP

- We translate the ASP program to *convenient* (reasonable size) SAT program and employ off-the-self independent support calculator.

Implementation Details

IS for ASP

- We translate the ASP program to *convenient* (reasonable size) SAT program and employ off-the-self independent support calculator.
- For normal problem, we rely on standard translation from ASP to SAT.

Implementation Details

IS for ASP

- We translate the ASP program to *convenient* (reasonable size) SAT program and employ off-the-self independent support calculator.
- For normal problem, we rely on standard translation from ASP to SAT.
- For disjunctive problem, we translate to a SAT program F such that $\forall \tau \in \mathsf{AS}(P) \rightarrow \tau \in \mathsf{Sol}(F)$, where P is the disjunctive program.

Table of Content

Introduction

Preliminaries

Approximate Counting Technique

Implementation Details

Experiments

Experiments

Instances Description

Problem Class	Encoding Name	#Instances	Source
Normal	Vertex cover	1500	Randomly generated instances
	Independent set		
	Dominating set		
	Graph Reachability		
	r-arborescence		
Disjunctive	2QBF	200	

Experiments

Baseline & Parameters

Baseline

- Clingo
- DynASP
- ApproxMC
- Ganak

Parameters

- $\epsilon = 0.8$ and $\delta = 0.2$
- Timeout $=$ 5000 sec

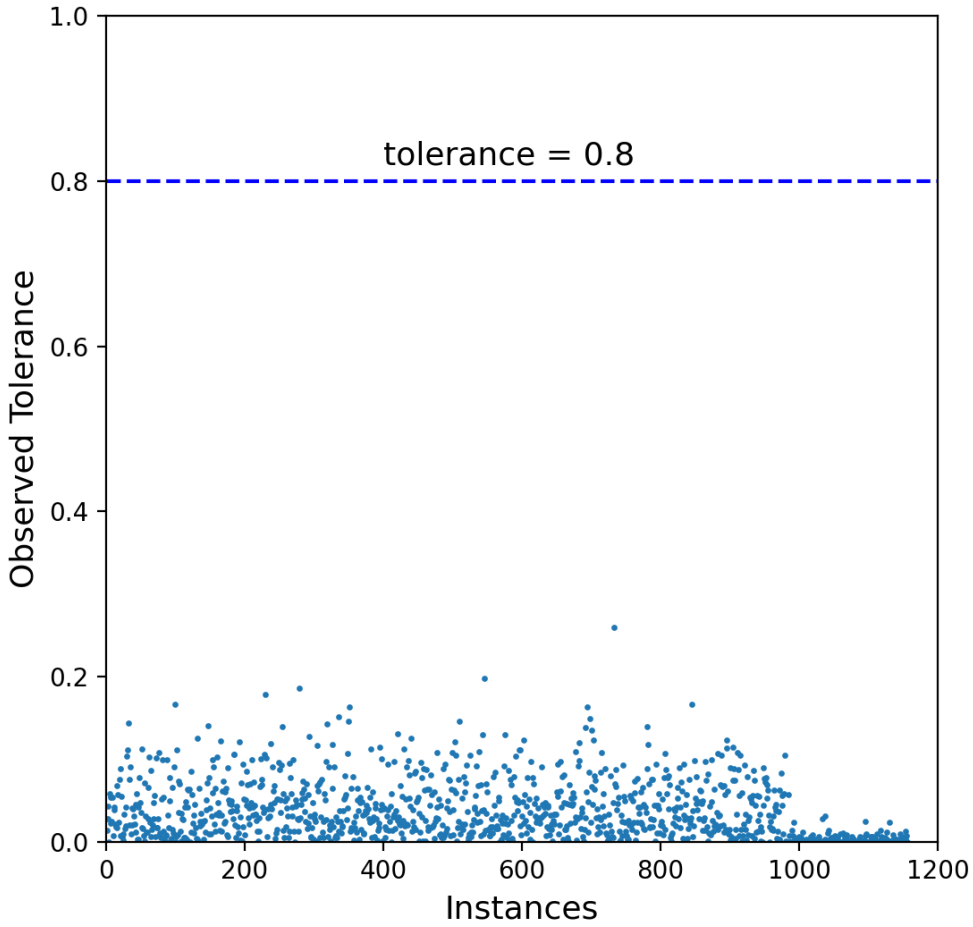
Experiments

Results

		Clingo	DynASP	Ganak	ApproxMC	ApproxASP
Normal	#Solved	738	47	973	**1325**	1323
	PAR-2	5172	9705	3606	**1200**	1218
Disjun.	#Solved	177	0	0	0	**185**
	PAR-2	1372	T	T	T	**795**

Experiments

Approximation Quality



$$\text{Observed tolerance} = \max\left(\frac{|\,\text{AS}(I)|}{\text{cnt}}, \frac{\text{cnt}}{|\,\text{AS}(I)|}\right) - 1$$

Conclusion

- We present ApproxASP – the first approximate answer set counter.
- ApproxASP solves instances that lie beyond the capability of off-the-self counting tools.

ApproxASP code

Alrabbaa, Christian, Sebastian Rudolph, and Lukas Schweizer (2018). "Faceted answer-set navigation". In: *International Joint Conference on Rules and Reasoning*. Springer, pp. 211–225.

Aziz, Rehan Abdul et al. (2015). "Stable model counting and its application in probabilistic logic programming". In: *Twenty-ninth AAAI conference on artificial intelligence*.

Carter, J Lawrence and Mark N Wegman (1977). "Universal classes of hash functions". In: *Journal of computer and system sciences* 18.2, pp. 143–154.

Fichte, Johannes Klaus, Sarah Alice Gaggl, and Dominik Rusovac (2022). "Rushing and Strolling among Answer Sets - Navigation Made Easy". In: *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI 2022)*.

Fichte, Johannes Klaus et al. (2016). "Counting Answer Sets via Dynamic Programming". In: *CoRR* abs/1612.07601.

Gebser, Martin, Benjamin Kaufmann, and Torsten Schaub (2012). "Conflict-driven answer set solving: From theory to practice". In: *Artificial Intelligence* 187, pp. 52–89.

Gebser, Martin et al. (2007). "clasp: A Conflict-Driven Answer Set Solver". In: *Logic Programming and Nonmonotonic Reasoning*. Ed. by Chitta Baral, Gerhard Brewka, and John Schlipf. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 260–265.

Han, Cheng-Shen and Jie-Hong Roland Jiang (2012). "When boolean satisfiability meets gaussian elimination in a simplex way". In: *International Conference on Computer Aided Verification*. Springer, pp. 410–426.

Hecher, Markus (Sept. 2020). "Treewidth-aware Reductions of Normal ASP to SAT - Is Normal ASP Harder than SAT after All?" In: *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning*, pp. 485–495.

Lee, Joohyung, Samidh Talsania, and Yi Wang (2017). "Computing LPMLN using ASP and MLN solvers". In: 17.5-6, pp. 942–960.

Soos, Mate, Stephan Gocht, and Kuldeep S Meel (2020). "Tinted, detached, and lazy CNF-XOR solving and its applications to counting and sampling". In: *International Conference on Computer Aided Verification*. Springer, pp. 463–484.

Stockmeyer, Larry (1983). "The complexity of approximate counting". In: *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pp. 118–126.