Optimization Problems in Answer Set Programming

Mario Alviano

Department of Mathematics and Computer Science
University of Calabria, Italy

Klagenfurt, Austria
29 May 2015

Outline

The talk has two parts

1. An overview on Answer Set Programming (ASP)
2. Solving optimization problems in ASP

Part I

An overview on Answer Set Programming (ASP)

Outline: Part I

1. Answer Set Programming
   - Informal introduction
   - Stable model semantics
   - Computational problems and complexity

2. ASP systems
   - Architecture of ASP systems
   - Learning

Outline: Part I

1 Answer Set Programming
- Informal introduction
- Stable model semantics
- Computational problems and complexity

2 ASP systems
- Architecture of ASP systems
- Learning

Outline: Part I

1 Answer Set Programming
- Informal introduction
- Stable model semantics
- Computational problems and complexity

2 ASP systems
- Architecture of ASP systems
- Learning

Answer Set Programming

What

Logic programming under stable model semantics

Where

- Consistent query answering
- Data integration
- Team-building
- Automatic allotment of package tours
- Inconsistencies detection in biological networks
- Planning under incomplete knowledge

The language

- Knowledge is encoded by rules of the form

$$\underbrace{\alpha_1 \mid \cdots \mid \alpha_m}_{\textit{head}} \text{ :- } \underbrace{\beta_1, \ldots, \beta_n}_{\textit{body}} \ .$$

head
a disjunction of atoms

body
a conjunction of literals

The language

- Knowledge is encoded by rules of the form

$$\underbrace{\alpha_1 \mid \cdots \mid \alpha_m}_{\textit{head}} \text{ :- } \underbrace{\beta_1, \ldots, \beta_n}_{\textit{body}} \ .$$

a disjunction of atoms

a conjunction of literals

- Default negation may occur in rule bodies

The language

- Knowledge is encoded by rules of the form

$$\underbrace{\alpha_1 \mid \cdots \mid \alpha_m}_{\textit{head}} \text{ :- } \underbrace{\beta_1, \ldots, \beta_n}_{\textit{body}} \ .$$

head: a disjunction of atoms

body: a conjunction of literals

- Default negation may occur in rule bodies
- Literals also include aggregates

Instantiation of aggregates

- Aggregates express properties involving sets of atoms

Example

pass(X) :– student(X), #sum{S,E : mark(X,E,S)} $\geq$ 60.
student(alice). student(bob).
mark(alice,ex_1,40). mark(alice,ex_2,40).
mark(bob,ex_1,50). mark(bob,ex_2,0).

Instantiation of aggregates

- Aggregates express properties involving sets of atoms

Example

```
pass(X) :– student(X), #sum{S,E : mark(X,E,S)} ≥ 60.
student(alice). student(bob).
mark(alice,ex_1,40). mark(alice,ex_2,40).
mark(bob,ex_1,50).   mark(bob,ex_2,0).
```

```
pass(alice) :– student(alice), #sum{40,ex_1 : mark(alice,ex_1,40);
      40,ex_2 : mark(alice,ex_2,40)} ≥ 60.
```

Instantiation of aggregates

- Aggregates express properties involving sets of atoms

Example

```
pass(X) :– student(X), #sum{S,E : mark(X,E,S)} ≥ 60.
student(alice). student(bob).
mark(alice,ex_1,40). mark(alice,ex_2,40).
mark(bob,ex_1,50).   mark(bob,ex_2,0).
```

```
pass(alice) :– student(alice), #sum{40,ex_1 : mark(alice,ex_1,40);
      40,ex_2 : mark(alice,ex_2,40)} ≥ 60.
pass(bob) :– student(bob), #sum{50,ex_1 : mark(bob,ex_1,50);
      0,ex_2 : mark(bob,ex_2,0)} ≥ 60.
```

Instantiation of aggregates

- Aggregates express properties involving sets of atoms

Example

```
pass(X) :– student(X), #sum{S,E : mark(X,E,S)} ≥ 60.
student(alice). student(bob).
mark(alice,ex_1,40). mark(alice,ex_2,40).
mark(bob,ex_1,50).   mark(bob,ex_2,0).
```

```
pass(alice) :– student(alice), #sum{40,ex_1 : mark(alice,ex_1,40);
      40,ex_2 : mark(alice,ex_2,40)} ≥ 60.
pass(bob) :– student(bob), #sum{50,ex_1 : mark(bob,ex_1,50);
      0,ex_2 : mark(bob,ex_2,0)} ≥ 60.
```

Actual result

```
pass(alice).
```

Outline: Part I

1 Answer Set Programming
- Informal introduction
- Stable model semantics
- Computational problems and complexity

2 ASP systems
- Architecture of ASP systems
- Learning

Stable models

- Intelligent grounding outputs a propositional program Π
- Interpretations and models are defined as usual

Stable models

- Intelligent grounding outputs a propositional program Π
- Interpretations and models are defined as usual
- The reduct Π^I of Π wrt an interpretation I is obtained by
  1. deleting rules with false bodies
  2. deleting negated literals

Stable models

- Intelligent grounding outputs a propositional program Π
- Interpretations and models are defined as usual
- The reduct Π^I of Π wrt an interpretation I is obtained by
  1. deleting rules with false bodies
  2. deleting negated literals
- I is a stable model if it is a minimal model of Π^I

Example

```
a :– not not a.
b :– c.
c :– b.
c :– a.
```

Stable models

1. {a, b, c}
2. $\emptyset$

Stable models

- Intelligent grounding outputs a propositional program Π
- Interpretations and models are defined as usual
- The reduct Π^I of Π wrt an interpretation I is obtained by
  1. deleting rules with false bodies
  2. deleting negated literals
- I is a stable model if it is a minimal model of Π^I

Example

```
a :– not not a.
b :– c.
c :– b.
c :– a.
```

Stable models

1. {a, b, c}
2. $\emptyset$

A program may have zero or several stable models!

Optimum stable models

- ASP optimization statements, or weak constraints

$$:\sim \beta_1, \ldots, \beta_n. \text{ [weight@level]}$$

- Stable models are assigned a cost
- Optimum stable models are stable models of minimum cost

Optimum stable models

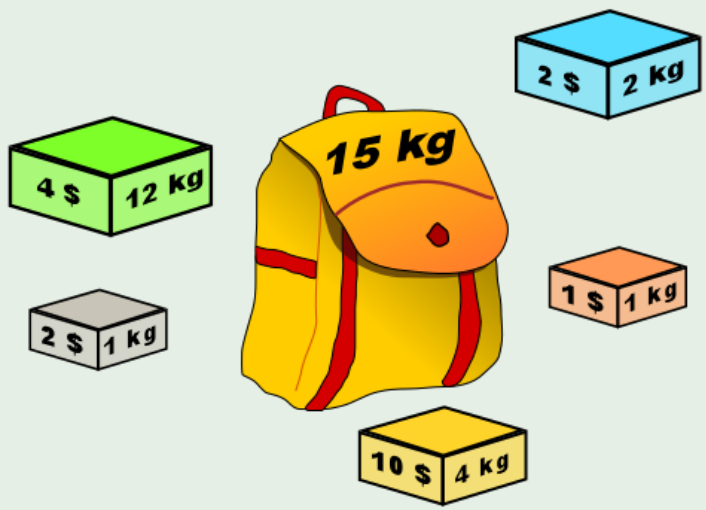
- ASP optimization statements, or weak constraints

$$:\sim \beta_1, \ldots, \beta_n. \text{ [weight@level]}$$

- Stable models are assigned a cost
- Optimum stable models are stable models of minimum cost

Example (Knapsack for lazy thief)

1. Stole as much value as possible
2. Minimize the total weight



Optimum stable models

- ASP optimization statements, or weak constraints

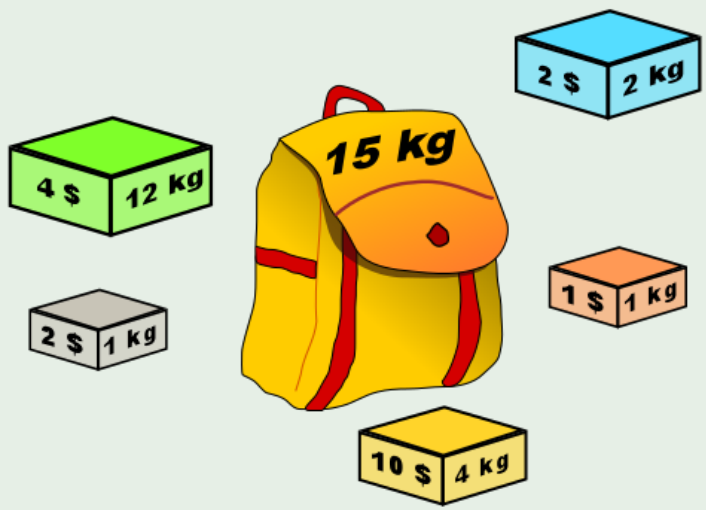
$$:\sim \beta_1, \ldots, \beta_n. \ [\text{weight@level}]$$

- Stable models are assigned a cost
- Optimum stable models are stable models of minimum cost

Example (Knapsack for lazy thief)

1. Stole as much value as possible
2. Minimize the total weight

```
object(green). ...
value(green,4). ...
weight(green,12). ...
```



Optimum stable models

- ASP optimization statements, or weak constraints

$$:\sim \beta_1, \ldots, \beta_n. \text{ [weight@level]}$$

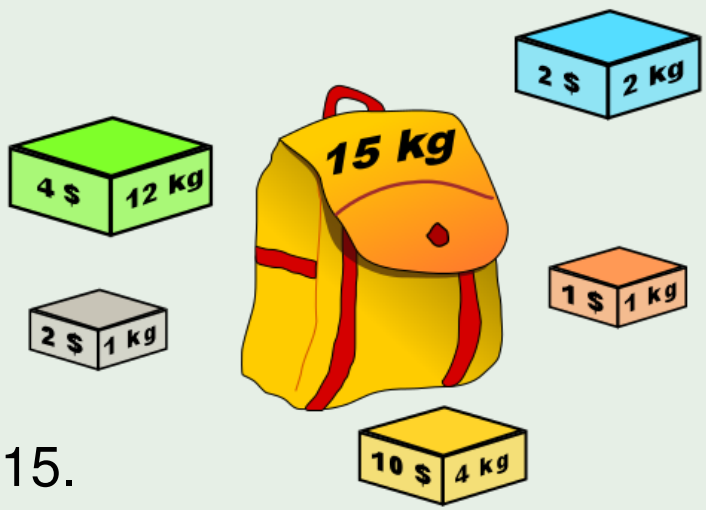
- Stable models are assigned a cost
- Optimum stable models are stable models of minimum cost

Example (Knapsack for lazy thief)

1. Stole as much value as possible
2. Minimize the total weight

```
object(green). ...
value(green,4). ...
weight(green,12). ...

in(X) :– object(X), not not in(X).
:– #sum{W,X : weight(X,W), in(X)} > 15.
:∼ value(X,V), not in(X). [V@2]
:∼ weight(X,W), in(X). [W@1]
```



Outline: Part I

1 Answer Set Programming
- Informal introduction
- Stable model semantics
- Computational problems and complexity

2 ASP systems
- Architecture of ASP systems
- Learning

Computational problems

- Model checking
- Coherence checking
- Brave reasoning
- Cautious reasoning
- Optimum stable model computation

Computational complexity

- Up to Δ_3^P in general

Problem	Complexity
Model checking	coNP
Coherence checking	Σ_2^P
Brave reasoning	Σ_2^P
Cautious reasoning	Π_2^P
Opt. stable model comp.	Δ_3^P

Computational complexity

- Up to Δ_3^P in general

Problem	Complexity	
Model checking	coNP	P
Coherence checking	Σ_2^P	NP
Brave reasoning	Σ_2^P	NP
Cautious reasoning	Π_2^P	coNP
Opt. stable model comp.	Δ_3^P	Δ_2^P

- Several fragments exist with different complexities
  - ✓ Head cycle free programs
  - ✓ Stratified non-convex aggregates

Computational complexity

- Up to Δ_3^P in general

Problem	Complexity			
Model checking	coNP		P	
Coherence checking	Σ_2^P		NP	
Brave reasoning	Σ_2^P		NP	
Cautious reasoning	Π_2^P		coNP	
Opt. stable model comp.	Δ_3^P	$\Delta_3^P[\log n]$	Δ_2^P	$\Delta_2^P[\log n]$

- Several fragments exist with different complexities
  - ✓ Head cycle free programs
  - ✓ Stratified non-convex aggregates
  - ✓ Fixed levels in weak constraints
  - ✓ Fixed weights in weak constraints

Outline: Part I

1 Answer Set Programming
- Informal introduction
- Stable model semantics
- Computational problems and complexity

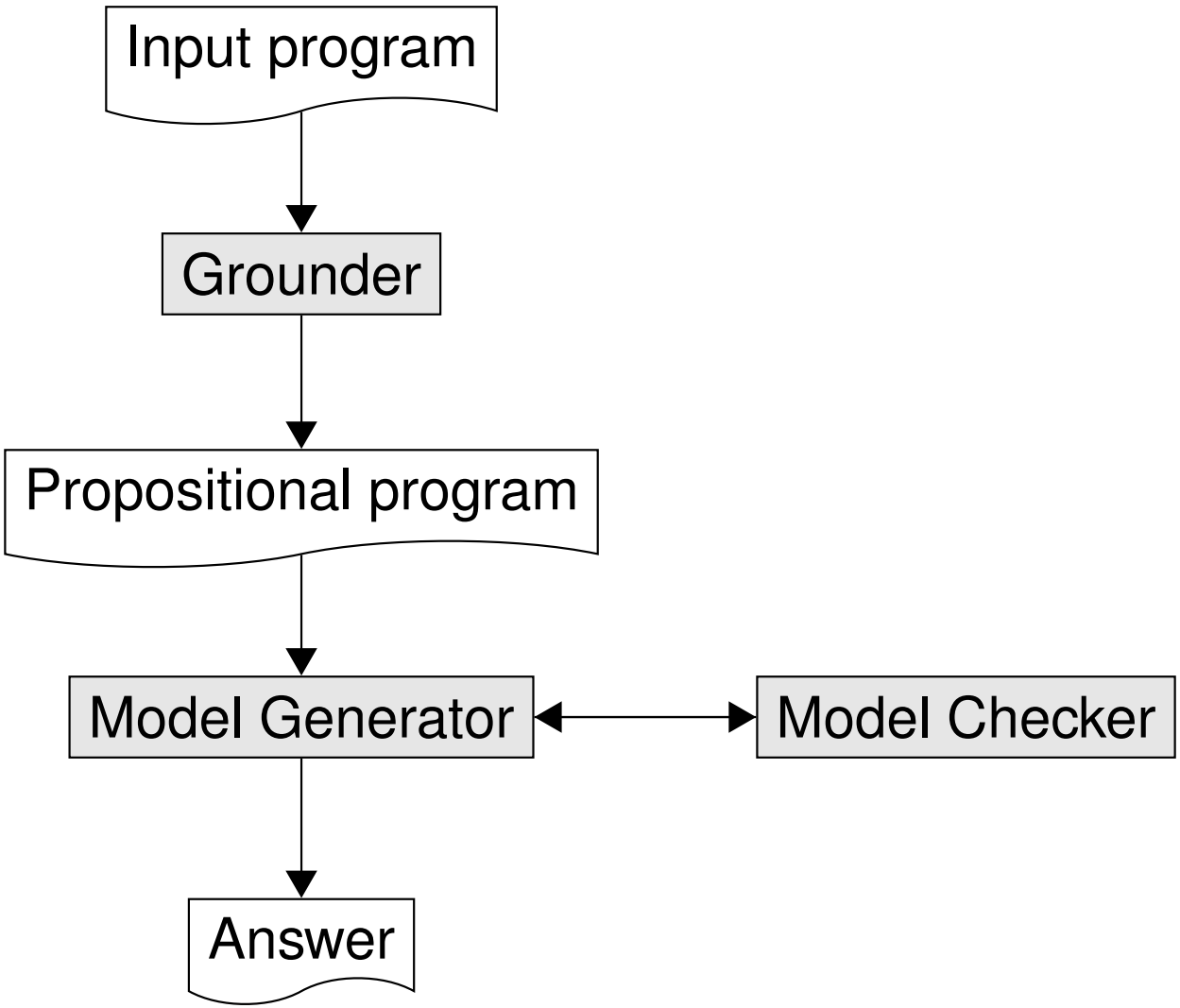
2 ASP systems
- Architecture of ASP systems
- Learning

Outline: Part I

1 Answer Set Programming
- Informal introduction
- Stable model semantics
- Computational problems and complexity

2 ASP systems
- Architecture of ASP systems
- Learning

Two phases process



Outline: Part I

1 Answer Set Programming
- Informal introduction
- Stable model semantics
- Computational problems and complexity

2 ASP systems
- Architecture of ASP systems
- Learning

A small example

Program Π

r_1: a :– not b.
r_2: c :– not d.
r_3: d :– a, c.
r_4: b :– not a.
r_5: d :– not c.
r_6: :– not a, b.

A small example

Program Π

r_1: a :– not b.
r_2: c :– not d.
r_3: d :– a, c.
r_4: b :– not a.
r_5: d :– not c.
r_6: :– not a, b.

Implication graph

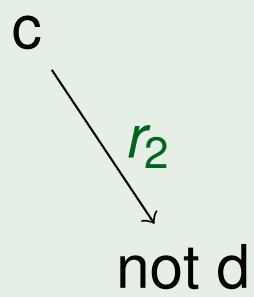
c

A small example

Program Π

r_1: a :– not b.
r_2: c :– not d.
r_3: d :– a, c.
r_4: b :– not a.
r_5: d :– not c.
r_6: :– not a, b.

Implication graph

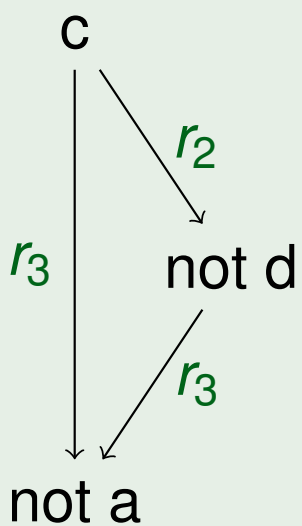


A small example

Program Π

r_1: a :– not b.
r_2: c :– not d.
r_3: d :– a, c.
r_4: b :– not a.
r_5: d :– not c.
r_6: :– not a, b.

Implication graph

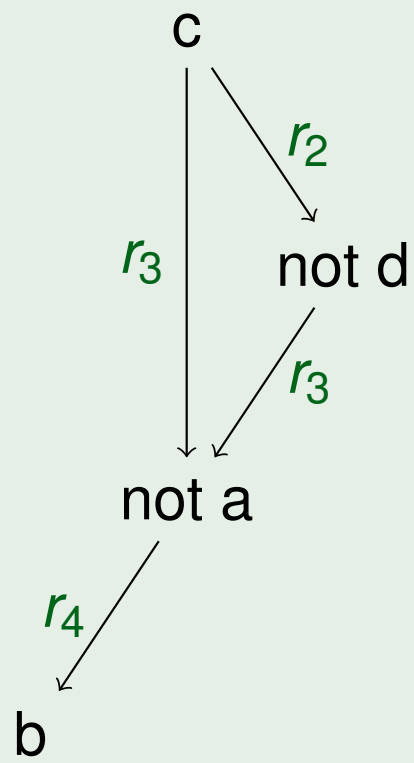


A small example

Program Π

r_1: a :– not b.
r_2: c :– not d.
r_3: d :– a, c.
r_4: b :– not a.
r_5: d :– not c.
r_6: :– not a, b.

Implication graph

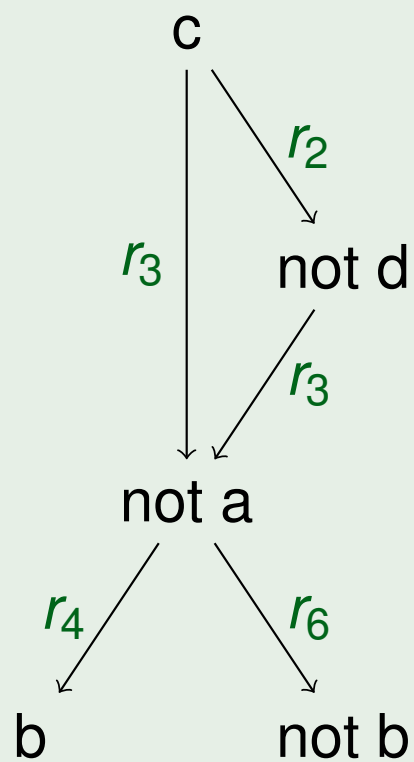


A small example

Program Π

r_1: a :– not b.
r_2: c :– not d.
r_3: d :– a, c.
r_4: b :– not a.
r_5: d :– not c.
r_6: :– not a, b.

Implication graph

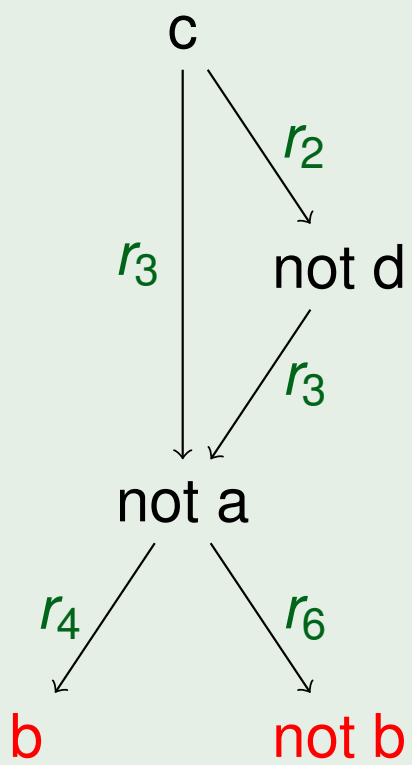


A small example

Program Π

r_1: a :– not b.
r_2: c :– not d.
r_3: d :– a, c.
r_4: b :– not a.
r_5: d :– not c.
r_6: :– not a, b.

Implication graph



Learning

Constraint

a

is implicit in Π
(no stable model
if a is false)

Part II

Solving optimization problems in ASP

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

Optimization problems in ASP

- Programs with weak constraints

$$:\sim \beta_1, \ldots, \beta_n.\ [\text{weight@level}]$$

Optimization problems in ASP

- Programs with weak constraints

$$:\sim \beta_1, \ldots, \beta_n.\ [\text{weight@level}]$$

- Normalized by adding soft literals

$$:\!- \beta_1, \ldots, \beta_n, \text{aux}.$$
$$:\sim \text{not aux}.\ [\text{weight@level}]$$
$$\text{aux} :\!- \text{not not aux}.$$

Optimization problems in ASP

- Programs with weak constraints

$$:\sim \beta_1, \ldots, \beta_n.\ [\text{weight@level}]$$

- Normalized by adding soft literals

$$:\!- \beta_1, \ldots, \beta_n, \text{aux}.$$
$$:\sim \text{not aux}.\ [\text{weight@level}]$$
$$\text{aux} :\!- \text{not not aux}.$$

- Weight and level are optional
- Omitted values are replaced by 1

Optimization problems in ASP

- Programs with weak constraints

$$:\sim \beta_1, \ldots, \beta_n.\ [\text{weight@level}]$$

- Normalized by adding soft literals

$$:\!- \beta_1, \ldots, \beta_n, \text{aux}.$$
$$:\sim \text{not aux}.\ [\text{weight@level}]$$
$$\text{aux} :\!- \text{not not aux}.$$

- Weight and level are optional
- Omitted values are replaced by 1

Let's first consider programs without weights and levels!

$\min n - |\{aux_i \mid i \in [1..n]\}|$ s.t. Π'

$\max |\{aux_i \mid i \in [1..n]\}| - n$ s.t. Π'

Running example

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
Π′		normalized	soft literals

Running example

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
Π′		normalized	soft literals

Associated optimization problem

$\min 3 - |\{a, b, c\}|$ s.t. Π'

$\max |\{a, b, c\}| - 3$ s.t. Π'

Running example

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
Π′		normalized	soft literals

Associated optimization problem

$$\min 3 - |\{a, b, c\}| \quad \text{s.t. } \Pi' \qquad\qquad \max |\{a, b, c\}| - 3 \quad \text{s.t. } \Pi'$$

- Three optimum stable models: $\{a\}$, $\{b\}$, and $\{c\}$
- The optimum cost is 2

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

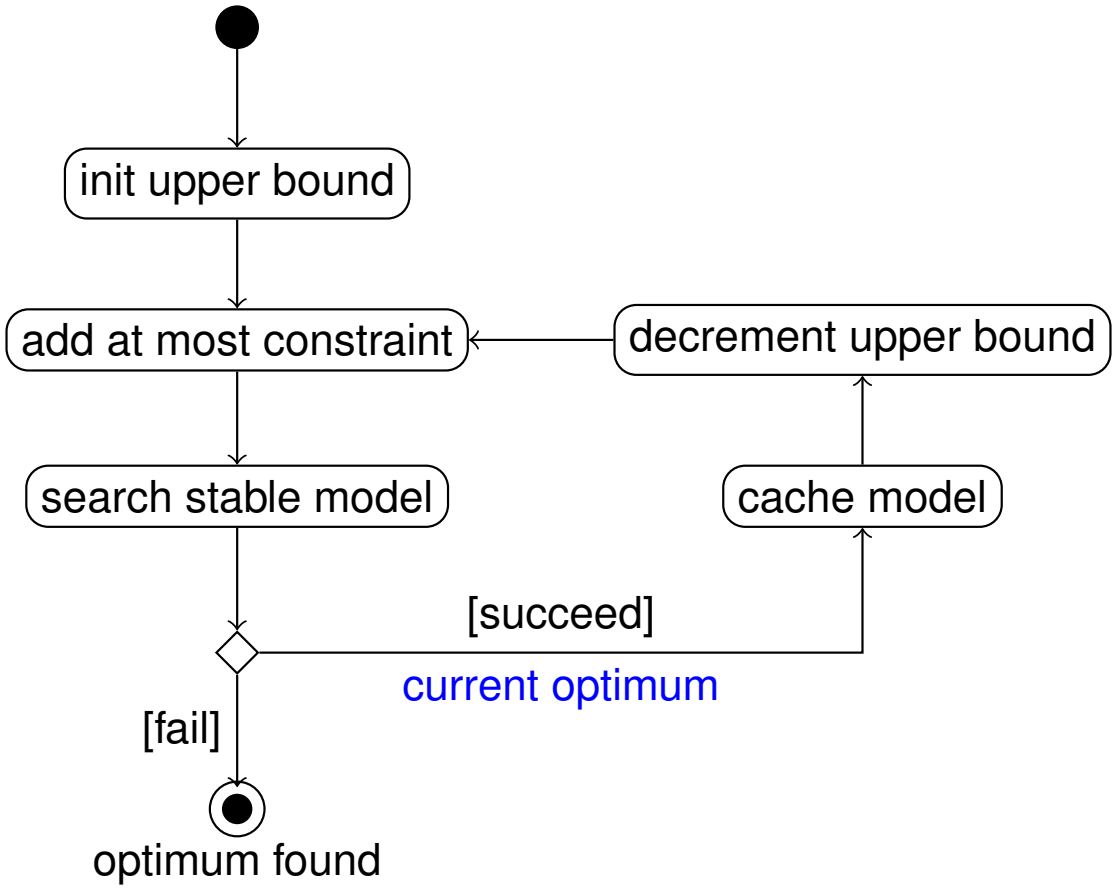
4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

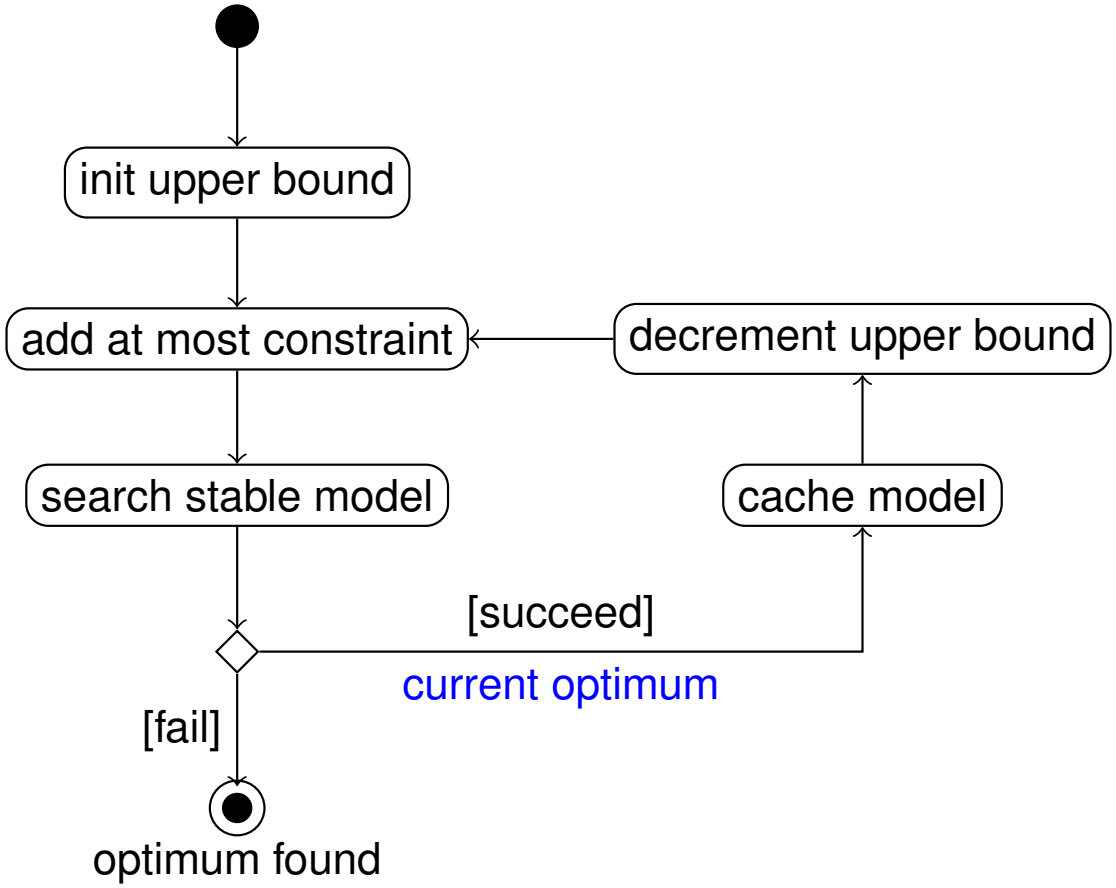
Branch and bound (1)

I'm hungry! Give me any stable model



Branch and bound (1)

I'm hungry! Give me any stable model



- Several variants: BB, OPT, MGD
- Poor performance on industrial instances

Branch and bound (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
Π′		normalized	soft literals

- Upper bound: 4

Search stable models of Π′ such that

1. $3 - a - b - c \leq 4 - 1 = 3$ (iff $a + b + c \geq 3 - 3 = 0$)

Branch and bound (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
Π′		normalized	soft literals

- Upper bound: ~~4~~ 3

Search stable models of Π′ such that

1. $3 - a - b - c \leq 4 - 1 = 3$ (iff $a + b + c \geq 3 - 3 = 0$)
   - ✓ $\emptyset$ of cost 3

Branch and bound (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
Π′		normalized	soft literals

- Upper bound: ~~4~~ 3

Search stable models of Π′ such that

1. $3 - a - b - c \leq 4 - 1 = 3$ (iff $a + b + c \geq 3 - 3 = 0$)
   - ✓ $\emptyset$ of cost 3
2. $3 - a - b - c \leq 3 - 1 = 2$ (iff $a + b + c \geq 3 - 2 = 1$)

Branch and bound (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
Π′		normalized	soft literals

- Upper bound: ~~4~~ ~~3~~ 2

Search stable models of Π′ such that

1. $3 - a - b - c \leq 4 - 1 = 3$ (iff $a + b + c \geq 3 - 3 = 0$)
   - ✓ $\emptyset$ of cost 3
2. $3 - a - b - c \leq 3 - 1 = 2$ (iff $a + b + c \geq 3 - 2 = 1$)
   - ✓ $\{a\}$ of cost 2

Branch and bound (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
Π′		normalized	soft literals

- Upper bound: ~~4~~ ~~3~~ 2

Search stable models of Π′ such that

1. $3 - a - b - c \leq 4 - 1 = 3$ (iff $a + b + c \geq 3 - 3 = 0$)
   - ✓ $\emptyset$ of cost 3
2. $3 - a - b - c \leq 3 - 1 = 2$ (iff $a + b + c \geq 3 - 2 = 1$)
   - ✓ $\{a\}$ of cost 2
3. $3 - a - b - c \leq 2 - 1 = 1$ (iff $a + b + c \geq 3 - 1 = 2$)

Branch and bound (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
Π′		normalized	soft literals

- Upper bound: ~~4~~ ~~3~~ 2

Search stable models of Π′ such that

1. $3 - a - b - c \leq 4 - 1 = 3$ (iff $a + b + c \geq 3 - 3 = 0$)
   - ✓ $\emptyset$ of cost 3
2. $3 - a - b - c \leq 3 - 1 = 2$ (iff $a + b + c \geq 3 - 2 = 1$)
   - ✓ $\{a\}$ of cost 2
3. $3 - a - b - c \leq 2 - 1 = 1$ (iff $a + b + c \geq 3 - 1 = 2$)
   - ✗ incoherent

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

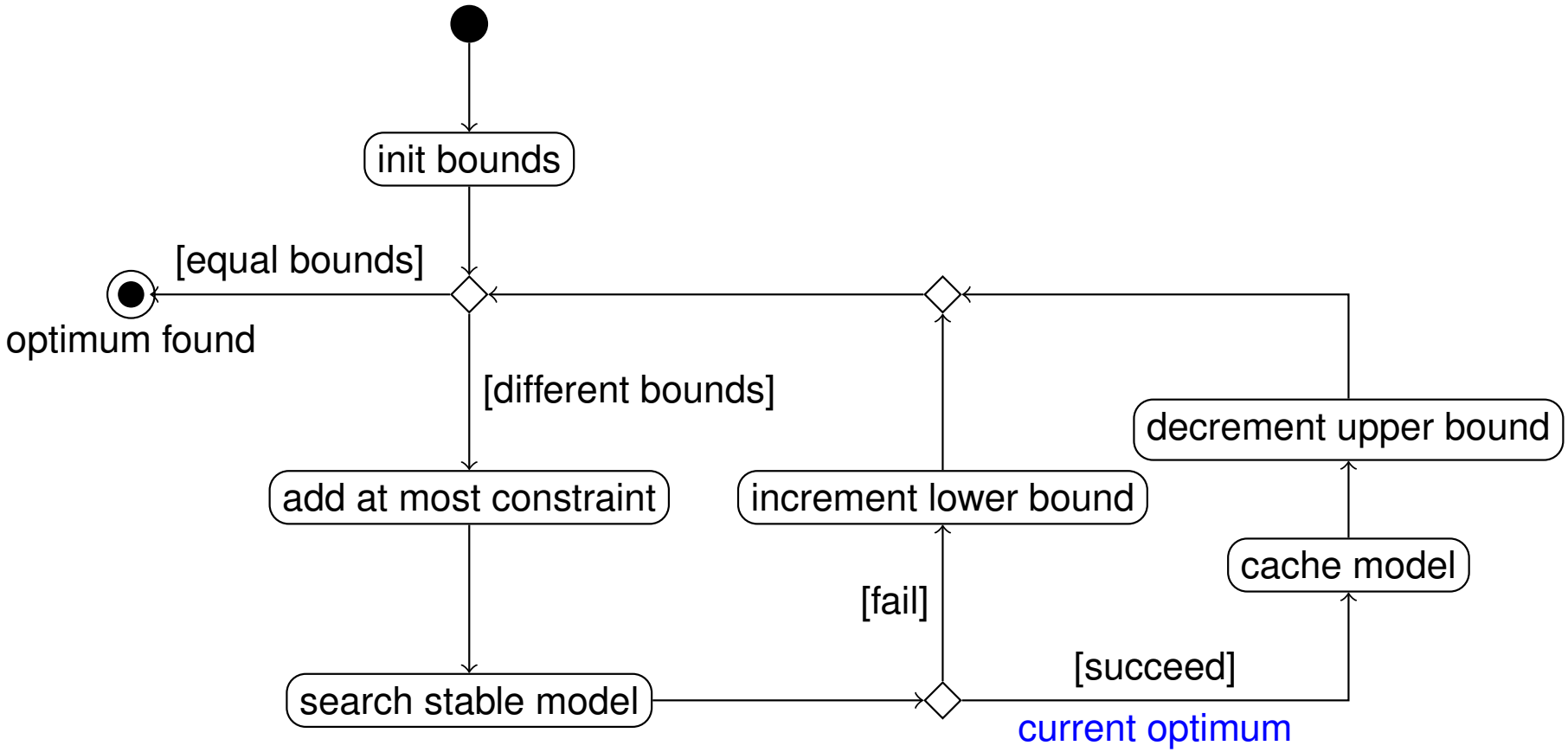
4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

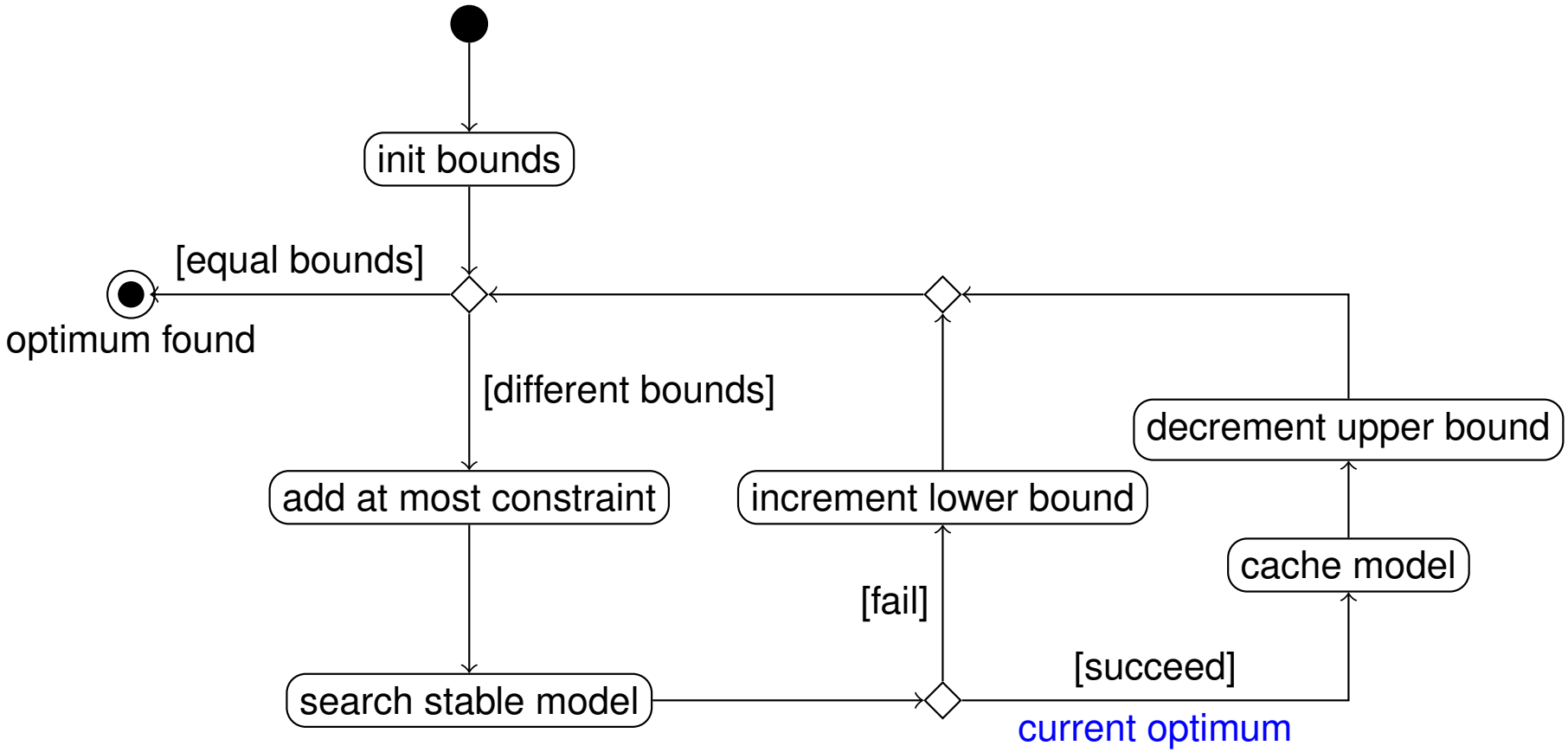
Binary search (1)

I believe (blindly) in the theory!
Complexity is $\Delta_3^P[\log n]$ or $\Delta_2^P[\log n]$



Binary search (1)

I believe (blindly) in the theory!
Complexity is $\Delta_3^P[\log n]$ or $\Delta_2^P[\log n]$



- Poor performance in practice

Binary search (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
Π′		normalized	soft literals

- Lower bound: 0
- Upper bound: 4

Search stable models of Π′ such that

1. $3 - a - b - c \leq \lceil \frac{4+0}{2} \rceil = 2 \quad (\text{iff } a + b + c \geq 3 - 2 = 1)$

Binary search (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
Π′		normalized	soft literals

- Lower bound: 0
- Upper bound: ~~4~~ 2

Search stable models of Π′ such that

1. $3 - a - b - c \leq \lceil \frac{4+0}{2} \rceil = 2$ (iff $a + b + c \geq 3 - 2 = 1$)
   - ✓ $\{a\}$ of cost 2

Binary search (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
Π′		normalized	soft literals

- Lower bound: 0
- Upper bound: ~~4~~ 2

Search stable models of Π′ such that

1. $3 - a - b - c \leq \lceil \frac{4+0}{2} \rceil = 2$ (iff $a + b + c \geq 3 - 2 = 1$)
   - ✓ $\{a\}$ of cost 2
2. $3 - a - b - c \leq \lceil \frac{2+0}{2} \rceil = 1$ (iff $a + b + c \geq 3 - 1 = 2$)

Binary search (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
Π′		normalized	soft literals

- Lower bound: ~~0~~ 2
- Upper bound: ~~4~~ 2

Search stable models of Π′ such that

1. $3 - a - b - c \leq \lceil \frac{4+0}{2} \rceil = 2 \quad (\text{iff } a + b + c \geq 3 - 2 = 1)$
   - ✓ $\{a\}$ of cost 2
2. $3 - a - b - c \leq \lceil \frac{2+0}{2} \rceil = 1 \quad (\text{iff } a + b + c \geq 3 - 1 = 2)$
   - ✗ incoherent

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

Assumptions and unsatisfiable cores

Program Π

r_1: a :– not b.
r_2: c :– not d.
r_3: d :– a, c.
r_4: b :– not a.
r_5: d :– not c.
r_6: :– not a, b.

Assumptions

c, not a

Assumptions and unsatisfiable cores

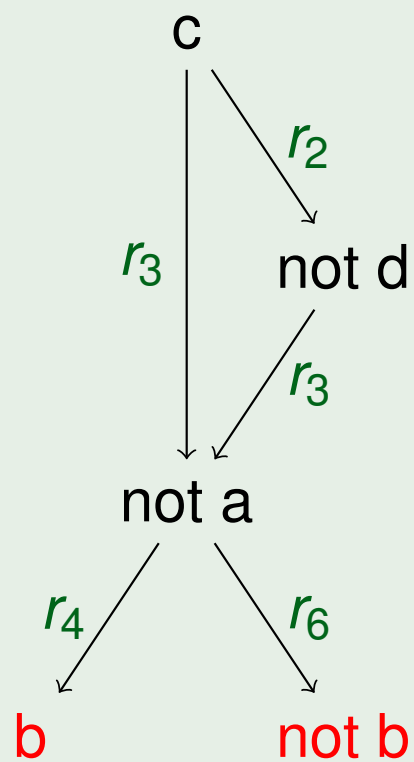
Program Π

r_1: a :– not b.
r_2: c :– not d.
r_3: d :– a, c.
r_4: b :– not a.
r_5: d :– not c.
r_6: :– not a, b.

Assumptions

c, not a

Implication graph



Learning

Constraint

a

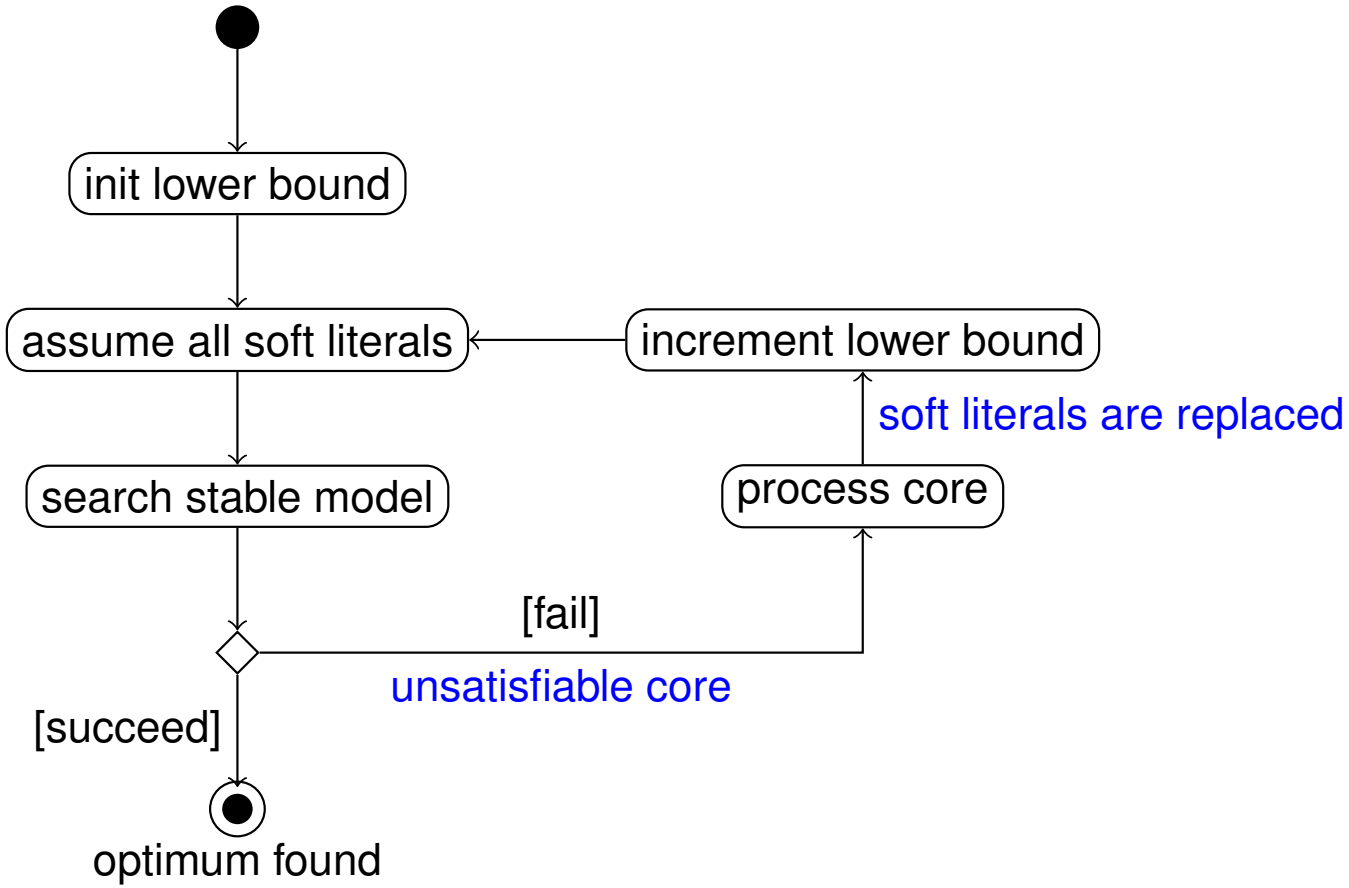
is implicit in Π
(no stable model
if a is false)

Unsatisfiable core

not a

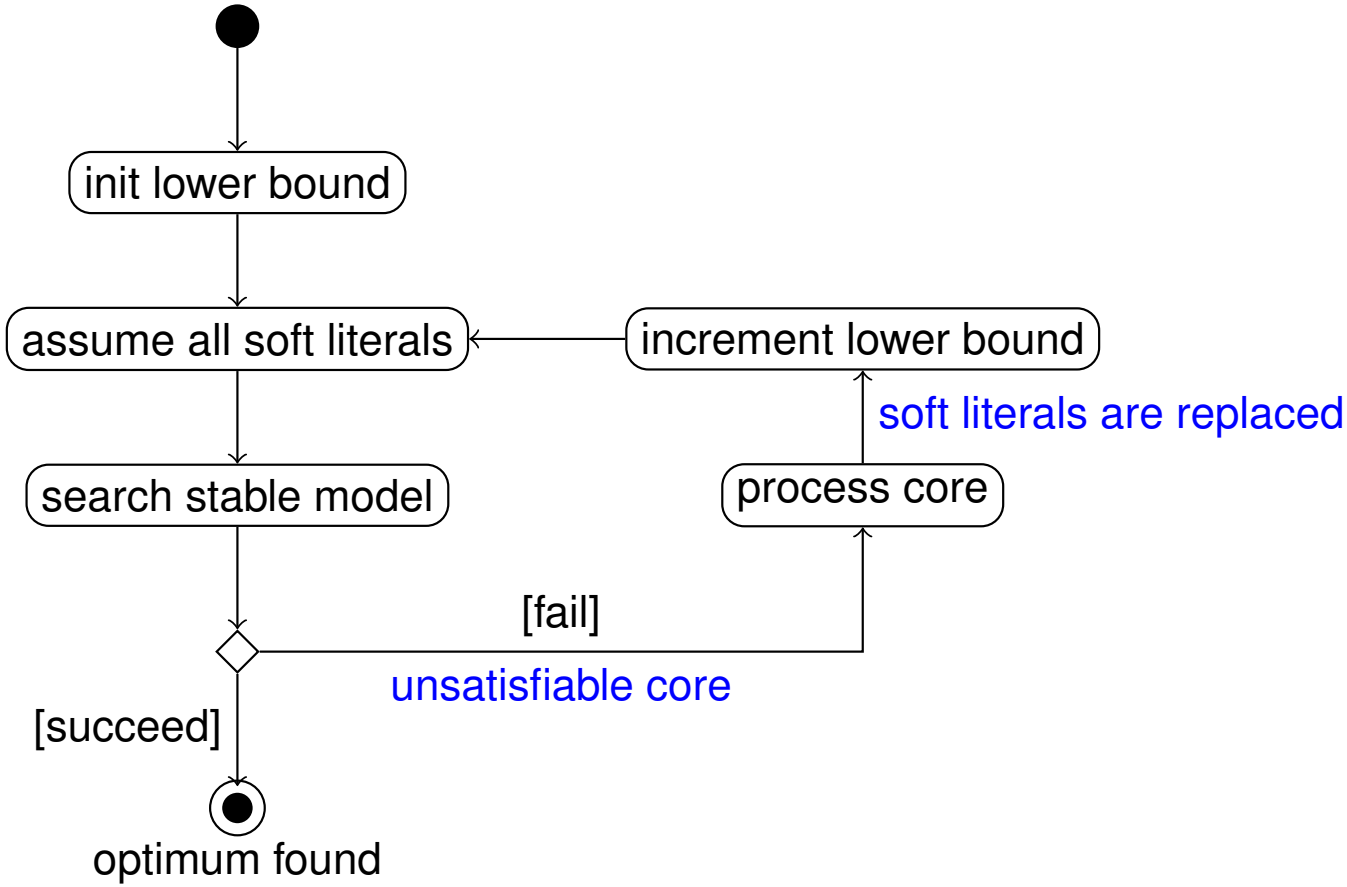
Core-based algorithms

I'm feeling lucky! Try to satisfy all soft literals



Core-based algorithms

I'm feeling lucky! Try to satisfy all soft literals



- Several strategies for processing cores
- Good performance on industrial instances

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d		

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$(a + b + d \geq 2) \vee \neg x_1$
1	c, x_1		

Program Π

a :– not not a. :– a, b. :∼ not a. (a,1)
b :– not not b. :– a, c. :∼ not b. (b,1)
c :– not not c. :– b, c. :∼ not c. (c,1)
d :– not not d. :– d. :∼ not d. (d,1)

Π′ | normalized | soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$(a + b + d \geq 2) \vee \neg x_1$
1	c, x_1	c, x_1	$(a + b + d \geq 1) \vee \neg x_2$
			$(c + x_1 \geq 1) \vee \neg x_3$
2	x_2, x_3		

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$(a + b + d \geq 2) \vee \neg x_1$
1	c, x_1	c, x_1	$(a + b + d \geq 1) \vee \neg x_2$
			$(c + x_1 \geq 1) \vee \neg x_3$
2	x_2, x_3	x_2, x_3	$(x_2 + x_3 \geq 1) \vee \neg x_4$
3	x_4		

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$(a + b + d \geq 2) \vee \neg x_1$
1	c, x_1	c, x_1	$(a + b + d \geq 1) \vee \neg x_2$
			$(c + x_1 \geq 1) \vee \neg x_3$
2	x_2, x_3	x_2, x_3	$(x_2 + x_3 \geq 1) \vee \neg x_4$
3	x_4	—	

Program Π

a :– not not a. :– a, b. :∼ not a. (a,1)
b :– not not b. :– a, c. :∼ not b. (b,1)
c :– not not c. :– b, c. :∼ not c. (c,1)
d :– not not d. :– d. :∼ not d. (d,1)

Π′ | normalized | soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$(a + b + d \geq 2) \vee \neg x_1$
1	c, x_1	c, x_1	$(a + b + d \geq 1) \vee \neg x_2$
			$(c + x_1 \geq 1) \vee \neg x_3$
2	x_2, x_3	x_2, x_3	$(x_2 + x_3 \geq 1) \vee \neg x_4$
3	x_4	—	

Optimum stable model: $\{a\} \cup \{x_2, x_4\}$

- Introduced in CLASP

- Introduced in CLASP
- ✓ Better than previous core-based algorithms

- Introduced in CLASP
- ✓ Better than previous core-based algorithms
- ✗ Several aggregates for the same core

OLL (2)

- ■ Introduced in CLASP
- ✓ Better than previous core-based algorithms
- ✗ Several aggregates for the same core
- ✓ Smart encoding with sorting networks in MSCG

- ■ Introduced in CLASP
- ✓ Better than previous core-based algorithms
- ✗ Several aggregates for the same core
- ✓ Smart encoding with sorting networks in MSCG
- ✗ Exponential blow up in the number of refutations

- Introduced in CLASP
- ✓ Better than previous core-based algorithms
- ✗ Several aggregates for the same core
- ✓ Smart encoding with sorting networks in MSCG
- ✗ Exponential blow up in the number of refutations

Blow up in refutations

$$x_1 + \cdots + x_n = b$$

b	satisfying assignments
$n-1$	

- Introduced in CLASP
- ✓ Better than previous core-based algorithms
- ✗ Several aggregates for the same core
- ✓ Smart encoding with sorting networks in MSCG
- ✗ Exponential blow up in the number of refutations

Blow up in refutations

$$x_1 + \cdots + x_n = b$$

b	satisfying assignments
$n-1$	$\binom{n}{n-1} = \frac{n!}{(n-1)!\cdot 1!}$

- ■ Introduced in CLASP
- ✓ Better than previous core-based algorithms
- ✗ Several aggregates for the same core
- ✓ Smart encoding with sorting networks in MSCG
- ✗ Exponential blow up in the number of refutations

Blow up in refutations

$$x_1 + \cdots + x_n = b$$

b	satisfying assignments	
$n-1$	$\binom{n}{n-1} = \frac{n!}{(n-1)!\cdot 1!}$	linear
$n-2$		

- ■ Introduced in CLASP
- ✓ Better than previous core-based algorithms
- ✗ Several aggregates for the same core
- ✓ Smart encoding with sorting networks in MSCG
- ✗ Exponential blow up in the number of refutations

Blow up in refutations

$$x_1 + \cdots + x_n = b$$

b	satisfying assignments	
$n-1$	$\binom{n}{n-1} = \frac{n!}{(n-1)!\cdot 1!}$	linear
$n-2$	$\binom{n}{n-2} = \frac{n!}{(n-2)!\cdot 2!}$	

- ■ Introduced in CLASP
- ✓ Better than previous core-based algorithms
- ✗ Several aggregates for the same core
- ✓ Smart encoding with sorting networks in MSCG
- ✗ Exponential blow up in the number of refutations

Blow up in refutations

$$x_1 + \cdots + x_n = b$$

b	satisfying assignments	
$n-1$	$\binom{n}{n-1} = \frac{n!}{(n-1)!\cdot 1!}$	linear
$n-2$	$\binom{n}{n-2} = \frac{n!}{(n-2)!\cdot 2!}$	quadratic
$n-3$		

- ■ Introduced in CLASP
- ✓ Better than previous core-based algorithms
- ✗ Several aggregates for the same core
- ✓ Smart encoding with sorting networks in MSCG
- ✗ Exponential blow up in the number of refutations

Blow up in refutations

$$x_1 + \cdots + x_n = b$$

b	satisfying assignments	
$n-1$	$\binom{n}{n-1} = \frac{n!}{(n-1)!\cdot 1!}$	linear
$n-2$	$\binom{n}{n-2} = \frac{n!}{(n-2)!\cdot 2!}$	quadratic
$n-3$	$\binom{n}{n-3} = \frac{n!}{(n-3)!\cdot 3!}$	cubic
$\cdots$		
$n/2$	$\binom{n}{n/2} = \frac{n!}{(n/2)!\cdot(n/2)!}$	

- Introduced in CLASP
- ✓ Better than previous core-based algorithms
- ✗ Several aggregates for the same core
- ✓ Smart encoding with sorting networks in MSCG
- ✗ Exponential blow up in the number of refutations

Blow up in refutations

$$x_1 + \cdots + x_n = b$$

b	satisfying assignments	
$n-1$	$\binom{n}{n-1} = \frac{n!}{(n-1)!\cdot 1!}$	linear
$n-2$	$\binom{n}{n-2} = \frac{n!}{(n-2)!\cdot 2!}$	quadratic
$n-3$	$\binom{n}{n-3} = \frac{n!}{(n-3)!\cdot 3!}$	cubic
$\ldots$		
$n/2$	$\binom{n}{n/2} = \frac{n!}{(n/2)!\cdot(n/2)!}$	exponential

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d		

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$a + b + d + \neg x_1 + \neg x_2 \geq 2$
			$x_1 \rightarrow x_2$
1	c, x_1, x_2		

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$a + b + d + \neg x_1 + \neg x_2 \geq 2$
			$x_1 \rightarrow x_2$
1	c, x_1, x_2	c, x_1	$c + x_1 + \neg x_3 \geq 1$
2	x_2, x_3		

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$a + b + d + \neg x_1 + \neg x_2 \geq 2$
			$x_1 \rightarrow x_2$
1	c, x_1, x_2	c, x_1	$c + x_1 + \neg x_3 \geq 1$
2	x_2, x_3	x_2, x_3	$x_2 + x_3 + \neg x_4 \geq 1$
3	x_4		

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$a + b + d + \neg x_1 + \neg x_2 \geq 2$
			$x_1 \rightarrow x_2$
1	c, x_1, x_2	c, x_1	$c + x_1 + \neg x_3 \geq 1$
2	x_2, x_3	x_2, x_3	$x_2 + x_3 + \neg x_4 \geq 1$
3	x_4	—	

Optimum stable model: $\{a\} \cup \{x_2, x_4\}$

- Introduced in MAXINO

- Introduced in MAXINO
- ✓ Always one aggregate per processed core

- Introduced in MAXINO
- ✓ Always one aggregate per processed core
- ✓ Simple implementation

- Introduced in MAXINO
- ✓ Always one aggregate per processed core
- ✓ Simple implementation
- ✓ Smart encodings are not necessary

- Introduced in MAXINO
- ✓ Always one aggregate per processed core
- ✓ Simple implementation
- ✓ Smart encodings are not necessary
- ✗ Exponential blow up in the number of refutations

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

MaxSAT resolution

$$\frac{(a,1) \qquad (b,1)}{(a \vee b,1) \quad (a \wedge b,1)}$$

MaxSAT resolution

$$\frac{(a,1) \qquad (b,1)}{(a \vee b,1) \quad (a \wedge b,1)}$$

Processing core $\{a, b, c, d\}$

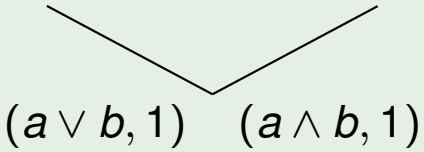
$(a,1)$ $(b,1)$ $(c,1)$ $(d,1)$

MaxSAT resolution

$$\frac{(a,1) \qquad (b,1)}{(a \vee b, 1) \quad (a \wedge b, 1)}$$

Processing core $\{a, b, c, d\}$

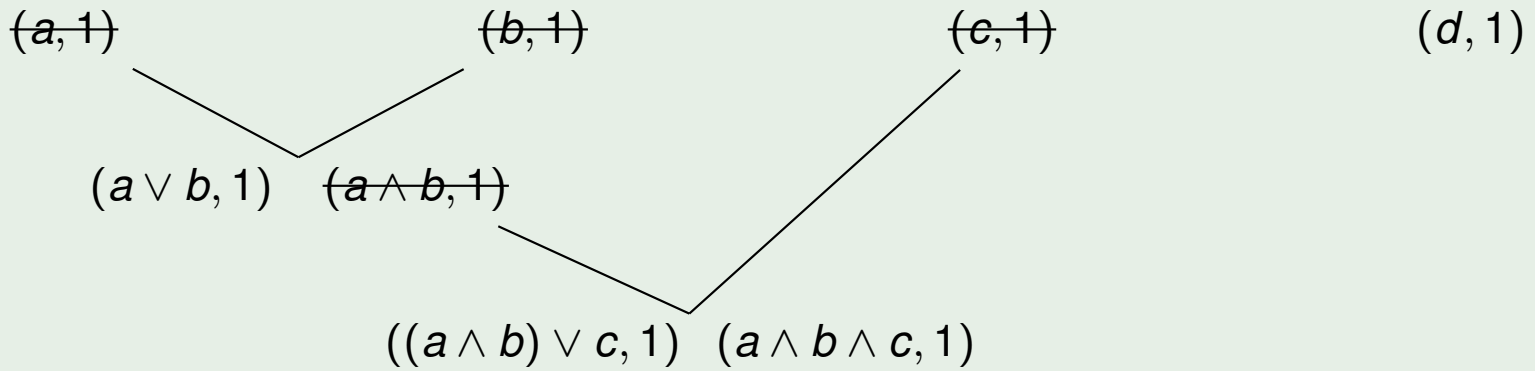
~~$(a,1)$~~ ~~$(b,1)$~~ $(c,1)$ $(d,1)$



MaxSAT resolution

$$\frac{(a,1) \qquad (b,1)}{(a \vee b, 1) \quad (a \wedge b, 1)}$$

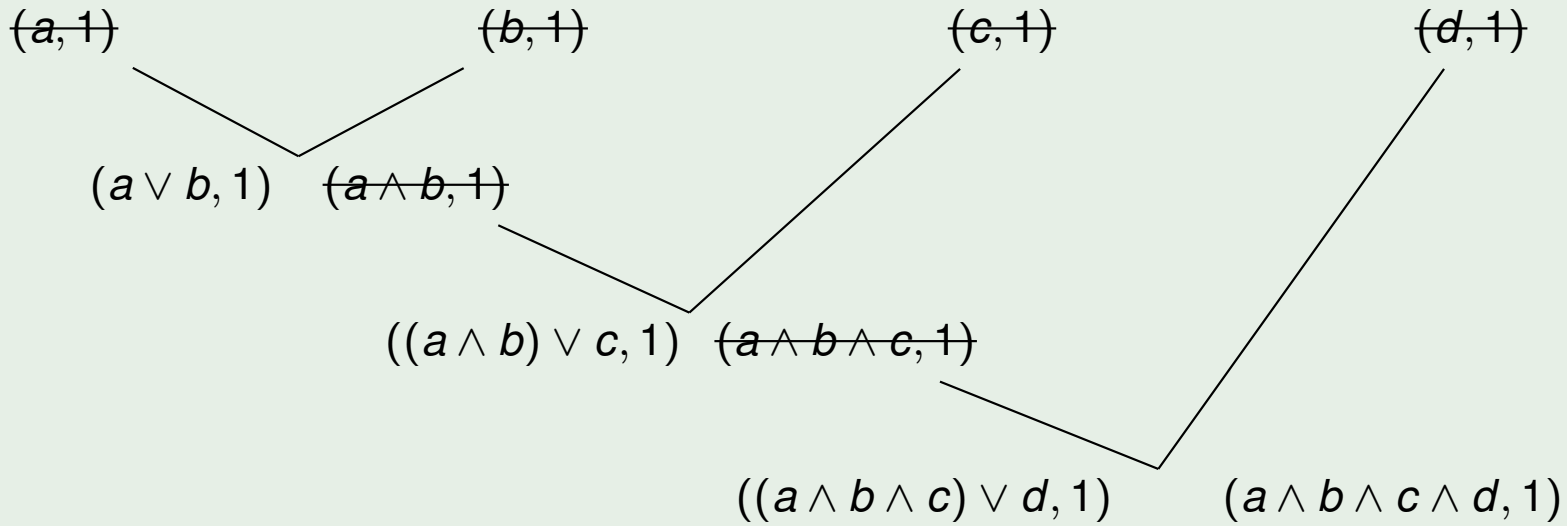
Processing core $\{a, b, c, d\}$



MaxSAT resolution

$$\frac{(a,1) \qquad (b,1)}{(a \vee b, 1) \quad (a \wedge b, 1)}$$

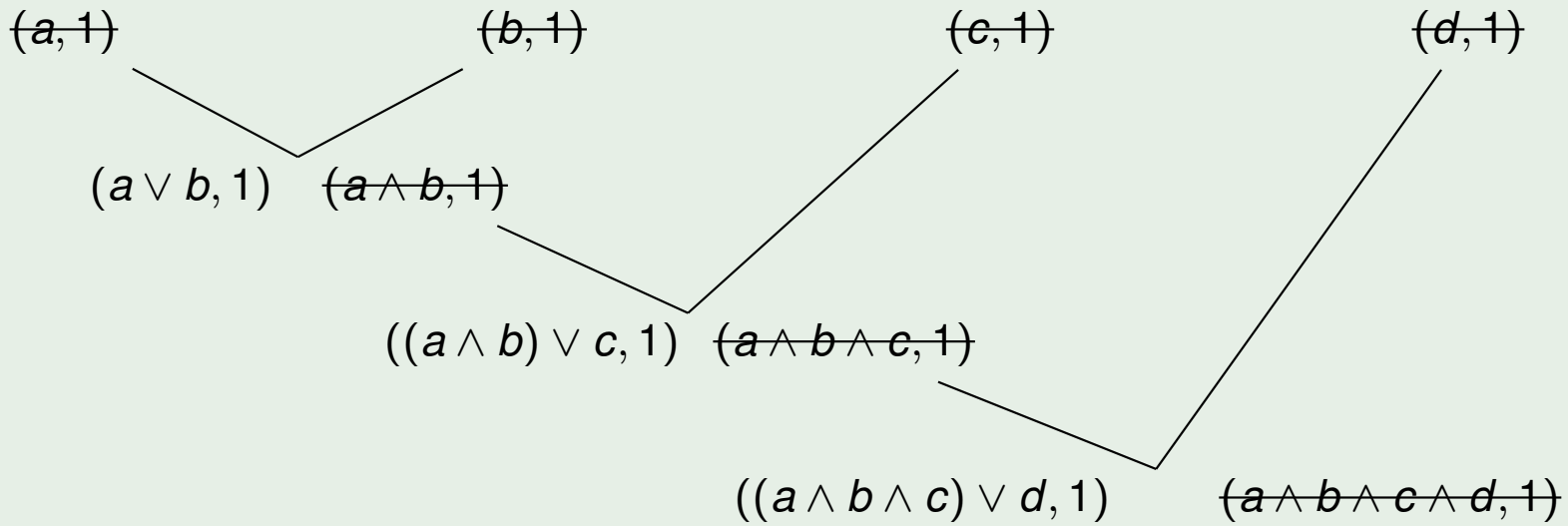
Processing core $\{a, b, c, d\}$



MaxSAT resolution

$$\frac{(a,1) \qquad (b,1)}{(a \vee b, 1) \quad (a \wedge b, 1)}$$

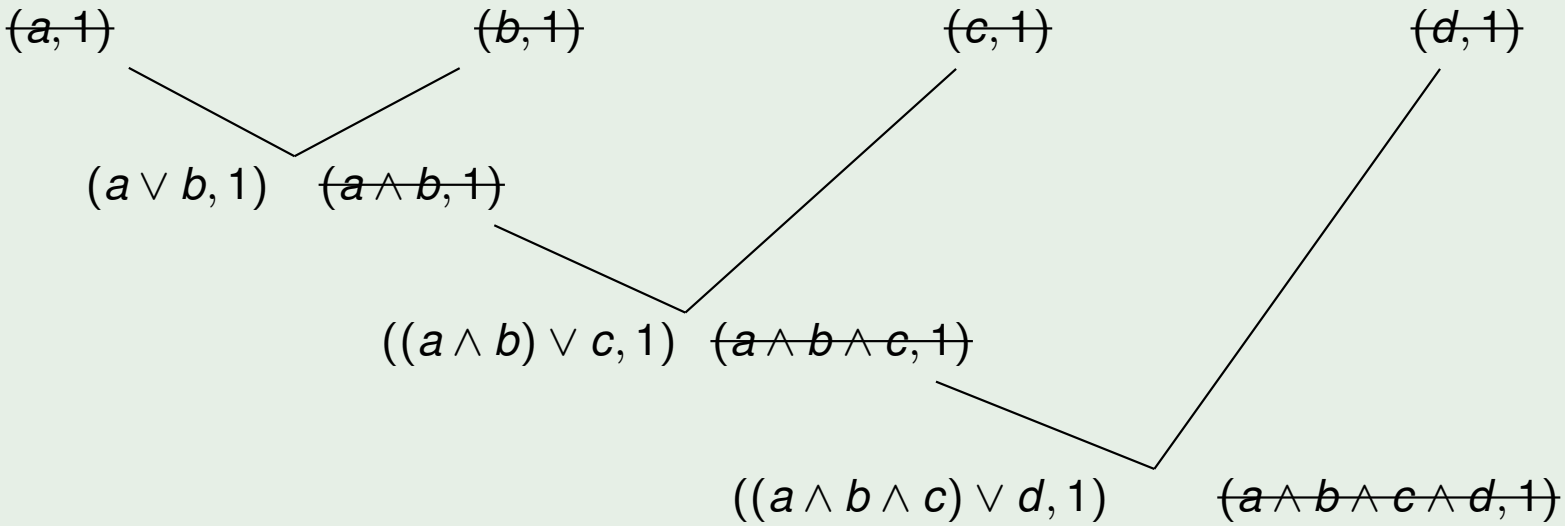
Processing core $\{a, b, c, d\}$



MaxSAT resolution

$$\frac{(a,1) \qquad (b,1)}{(a \vee b, 1) \quad (a \wedge b, 1)}$$

Processing core $\{a, b, c, d\}$



- New constraints
  $\neg x_1 \vee a \vee b \qquad y_1 \leftrightarrow a \wedge b$
  $\neg x_2 \vee y_1 \vee c \qquad y_2 \leftrightarrow y_1 \wedge c$
  $\neg x_3 \vee y_2 \vee d$
- New soft literals: x_1, x_2, and x_3

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d		

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$a \vee b \vee \neg x_1$
			$y_1 \vee d \vee \neg x_2$
			$y_1 \leftrightarrow a \wedge b$
1	c, x_1, x_2		

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$a \vee b \vee \neg x_1$
			$y_1 \vee d \vee \neg x_2$
			$y_1 \leftrightarrow a \wedge b$
1	c, x_1, x_2	c, x_1, x_2	$c \vee x_1 \vee \neg x_3$
			$y_2 \vee x_2 \vee \neg x_4$
			$y_2 \leftrightarrow c \wedge x_1$
2	x_3, x_4		

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$a \vee b \vee \neg x_1$
			$y_1 \vee d \vee \neg x_2$
			$y_1 \leftrightarrow a \wedge b$
1	c, x_1, x_2	c, x_1, x_2	$c \vee x_1 \vee \neg x_3$
			$y_2 \vee x_2 \vee \neg x_4$
			$y_2 \leftrightarrow c \wedge x_1$
2	x_3, x_4	x_3, x_4	$x_3 \vee x_4 \vee \neg x_5$
3	x_5		

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$a \vee b \vee \neg x_1$
			$y_1 \vee d \vee \neg x_2$
			$y_1 \leftrightarrow a \wedge b$
1	c, x_1, x_2	c, x_1, x_2	$c \vee x_1 \vee \neg x_3$
			$y_2 \vee x_2 \vee \neg x_4$
			$y_2 \leftrightarrow c \wedge x_1$
2	x_3, x_4	x_3, x_4	$x_3 \vee x_4 \vee \neg x_5$
3	x_5	—	

Program Π

a :– not not a.	:– a, b.	:∼ not a.	(a,1)
b :– not not b.	:– a, c.	:∼ not b.	(b,1)
c :– not not c.	:– b, c.	:∼ not c.	(c,1)
d :– not not d.	:– d.	:∼ not d.	(d,1)
Π′		normalized	soft literals

lb	soft literals	core	new constraints
0	a, b, c, d	a, b, d	$a \vee b \vee \neg x_1$
			$y_1 \vee d \vee \neg x_2$
			$y_1 \leftrightarrow a \wedge b$
1	c, x_1, x_2	c, x_1, x_2	$c \vee x_1 \vee \neg x_3$
			$y_2 \vee x_2 \vee \neg x_4$
			$y_2 \leftrightarrow c \wedge x_1$
2	x_3, x_4	x_3, x_4	$x_3 \vee x_4 \vee \neg x_5$
3	x_5	—	

Optimum stable model: $\{a\} \cup \{x_1, x_3, x_5\}$

- Introduced in EVA

- Introduced in EVA
- ✓ No exponential blow up in the number of refutations

- Introduced in EVA
- ✓ No exponential blow up in the number of refutations
- ✓ Only introduces clauses of bounded size

- Introduced in EVA
- ✓ No exponential blow up in the number of refutations
- ✓ Only introduces clauses of bounded size
- ✗ Several soft literals introduced for the same core may jointly belong to a new core

- Introduced in EVA
- ✓ No exponential blow up in the number of refutations
- ✓ Only introduces clauses of bounded size
- ✗ Several soft literals introduced for the same core may jointly belong to a new core
- ✗ Processing order may affect subsequent cores

- Introduced in EVA
- ✓ No exponential blow up in the number of refutations
- ✓ Only introduces clauses of bounded size
- ✗ Several soft literals introduced for the same core may jointly belong to a new core
- ✗ Processing order may affect subsequent cores
- ✗ Subsequent cores usually larger than ONE

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

From PMRES to K^4

Processing core $\{a, b, c, d\}$

From PMRES to K^4

Processing core $\{a, b, c, d\}$

PMRES

$$a \vee b \vee \neg x_1 \qquad y_1 \leftrightarrow a \wedge b$$
$$y_1 \vee c \vee \neg x_2 \qquad y_2 \leftrightarrow y_1 \wedge c$$
$$y_2 \vee d \vee \neg x_3$$

From PMRES to K^4

Processing core $\{a, b, c, d\}$

PMRES

$$a \vee b \vee \neg x_1 \qquad y_1 \leftrightarrow a \wedge b$$
$$y_1 \vee c \vee \neg x_2 \qquad y_2 \leftrightarrow y_1 \wedge c$$
$$y_2 \vee d \vee \neg x_3$$

PMRES-ish

$$a + b + \neg x_1 \geq 1 \qquad y_1 \leftrightarrow a \wedge b$$
$$y_1 + c + \neg x_2 \geq 1 \qquad y_2 \leftrightarrow y_1 \wedge c$$
$$y_2 + d + \neg x_3 \geq 1$$

From PMRES to K^4

Processing core $\{a, b, c, d\}$

PMRES

$$a \vee b \vee \neg x_1 \qquad y_1 \leftrightarrow a \wedge b$$
$$y_1 \vee c \vee \neg x_2 \qquad y_2 \leftrightarrow y_1 \wedge c$$
$$y_2 \vee d \vee \neg x_3$$

PMRES-ish

$$a + b + \neg x_1 \geq 1 \qquad y_1 \leftrightarrow a \wedge b$$
$$y_1 + c + \neg x_2 \geq 1 \qquad y_2 \leftrightarrow y_1 \wedge c$$
$$y_2 + d + \neg x_3 \geq 1$$

K^4

$$a + b + \neg y_1 + \neg x_1 \geq 2 \qquad y_1 \rightarrow x_1$$
$$y_1 + c + \neg y_2 + \neg x_2 \geq 2 \qquad y_2 \rightarrow x_2$$
$$y_2 + d \qquad + \neg x_3 \geq 1$$

From PMRES to K^4

Processing core $\{a, b, c, d\}$

PMRES

$$a \vee b \vee \neg x_1 \qquad y_1 \leftrightarrow a \wedge b$$
$$y_1 \vee c \vee \neg x_2 \qquad y_2 \leftrightarrow y_1 \wedge c$$
$$y_2 \vee d \vee \neg x_3$$

PMRES-ish

$$a + b + \neg x_1 \geq 1 \qquad y_1 \leftrightarrow a \wedge b$$
$$y_1 + c + \neg x_2 \geq 1 \qquad y_2 \leftrightarrow y_1 \wedge c$$
$$y_2 + d + \neg x_3 \geq 1$$

K^4

$$a + b + \neg y_1 + \neg x_1 \geq 2 \qquad y_1 \rightarrow x_1$$
$$y_1 + c + \neg y_2 + \neg x_2 \geq 2 \qquad y_2 \rightarrow x_2$$
$$y_2 + d \qquad + \neg x_3 \geq 1$$

- At most 4 literals in each constraint
- Number of satisfying assignments bounded by $\binom{4}{2} = 6$
- ✓ No exponential blow up!

From κ^4 to κ^6

- Let's add one more soft literal in each constraint

From κ^4 to κ^6

- Let's add one more soft literal in each constraint

κ^4

$$a + b + \neg y_1 + \neg x_1 \geq 2 \quad y_1 \rightarrow x_1$$
$$y_1 + c + \neg y_2 + \neg x_2 \geq 2 \quad y_2 \rightarrow x_2$$
$$y_2 + d + \neg x_3 \geq 1$$

ONE

$$a + b + c + d + \neg x_1 + \neg x_2 + \neg x_3 \geq 3 \quad x_1 \rightarrow x_2 \quad x_2 \rightarrow x_3$$

From κ^4 to κ^6

- Let's add one more soft literal in each constraint

κ^4

$$a + b + \neg y_1 + \neg x_1 \geq 2 \quad y_1 \rightarrow x_1$$
$$y_1 + c + \neg y_2 + \neg x_2 \geq 2 \quad y_2 \rightarrow x_2$$
$$y_2 + d + \neg x_3 \geq 1$$

ONE

$$a + b + c + d + \neg x_1 + \neg x_2 + \neg x_3 \geq 3 \quad x_1 \rightarrow x_2 \quad x_2 \rightarrow x_3$$

κ^6

$$a + b + c + \neg y_1 + \neg x_1 + \neg x_2 \geq 3 \quad y_1 \rightarrow x_1 \quad x_1 \rightarrow x_2$$
$$y_1 + d + \neg x_3 \geq 1$$

From κ^4 to κ^6

- Let's add one more soft literal in each constraint

κ^4

$$a + b + \neg y_1 + \neg x_1 \geq 2 \quad y_1 \rightarrow x_1$$
$$y_1 + c + \neg y_2 + \neg x_2 \geq 2 \quad y_2 \rightarrow x_2$$
$$y_2 + d + \neg x_3 \geq 1$$

ONE

$$a + b + c + d + \neg x_1 + \neg x_2 + \neg x_3 \geq 3 \quad x_1 \rightarrow x_2 \quad x_2 \rightarrow x_3$$

κ^6

$$a + b + c + \neg y_1 + \neg x_1 + \neg x_2 \geq 3 \quad y_1 \rightarrow x_1 \quad x_1 \rightarrow x_2$$
$$y_1 + d + \neg x_3 \geq 1$$

- At most 6 literals in each constraint
- Number of satisfying assignments bounded by $\binom{6}{3} = 20$

✓ No exponential blow up!

- In general, constraints of size $2 \cdot (k + 1)$, for any fixed k

✓ No exponential blow up!

- In general, constraints of size $2 \cdot (k+1)$, for any fixed k
- ✓ No exponential blow up!

Processing core $\{a_0, a_1, \ldots, a_8\}$

K^6

- In general, constraints of size $2 \cdot (k+1)$, for any fixed k
- ✓ No exponential blow up!

Processing core $\{a_0, a_1, \ldots, a_8\}$

K^6

$$a_0 + a_1 + a_2 + \neg y_1 + \neg x_1 + \neg x_2 \geq 3 \qquad y_1 \to x_1 \qquad x_1 \to x_2$$
$$y_1 + a_3 + a_4 + \neg y_2 + \neg x_3 + \neg x_4 \geq 3 \qquad y_2 \to x_3 \qquad x_3 \to x_4$$
$$y_2 + a_5 + a_6 + \neg y_3 + \neg x_5 + \neg x_6 \geq 3 \qquad y_3 \to x_5 \qquad x_5 \to x_6$$
$$y_3 + a_7 + a_8 \quad + \neg x_7 + \neg x_8 \geq 2 \qquad\qquad x_7 \to x_8$$

- In general, constraints of size $2 \cdot (k+1)$, for any fixed k
- ✓ No exponential blow up!

Processing core $\{a_0, a_1, \ldots, a_8\}$

K^6

$$a_0 + a_1 + a_2 + \neg y_1 + \neg x_1 + \neg x_2 \geq 3 \qquad y_1 \to x_1 \qquad x_1 \to x_2$$
$$y_1 + a_3 + a_4 + \neg y_2 + \neg x_3 + \neg x_4 \geq 3 \qquad y_2 \to x_3 \qquad x_3 \to x_4$$
$$y_2 + a_5 + a_6 + \neg y_3 + \neg x_5 + \neg x_6 \geq 3 \qquad y_3 \to x_5 \qquad x_5 \to x_6$$
$$y_3 + a_7 + a_8 + \neg x_7 + \neg x_8 \geq 2 \qquad x_7 \to x_8$$

K^{10} (symmetry breaker omitted)

$$a_0 + a_1 + a_2 + a_3 + a_4 + \neg y_1 + \neg x_1 + \neg x_2 + \neg x_3 + \neg x_4 \geq 5$$
$$y_1 + a_5 + a_6 + a_7 + a_8 + \neg x_5 + \neg x_6 + \neg x_7 + \neg x_8 \geq 4$$

- In general, constraints of size $2 \cdot (k+1)$, for any fixed k
- ✓ No exponential blow up!

Processing core $\{a_0, a_1, \ldots, a_8\}$

K^6

$$a_0 + a_1 + a_2 + \neg y_1 + \neg x_1 + \neg x_2 \geq 3 \quad y_1 \to x_1 \quad x_1 \to x_2$$
$$y_1 + a_3 + a_4 + \neg y_2 + \neg x_3 + \neg x_4 \geq 3 \quad y_2 \to x_3 \quad x_3 \to x_4$$
$$y_2 + a_5 + a_6 + \neg y_3 + \neg x_5 + \neg x_6 \geq 3 \quad y_3 \to x_5 \quad x_5 \to x_6$$
$$y_3 + a_7 + a_8 + \neg x_7 + \neg x_8 \geq 2 \quad x_7 \to x_8$$

K^{10} (symmetry breaker omitted)

$$a_0 + a_1 + a_2 + a_3 + a_4 + \neg y_1 + \neg x_1 + \neg x_2 + \neg x_3 + \neg x_4 \geq 5$$
$$y_1 + a_5 + a_6 + a_7 + a_8 + \neg x_5 + \neg x_6 + \neg x_7 + \neg x_8 \geq 4$$

K^{18} (symmetry breaker omitted)

$$a_0 + a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7 + a_8 + \neg x_1 + \neg x_2 + \neg x_3 + \neg x_4 + \neg x_5 + \neg x_6 + \neg x_7 + \neg x_8 \geq 8$$

- Introduced in MAXINO

- Introduced in MAXINO
- ✓ Extends both ONE and PMRES

- Introduced in MAXINO
- ✓ Extends both ONE and PMRES
- ✓ No exponential blow up in the number of refutations

- Introduced in MAXINO
- ✓ Extends both ONE and PMRES
- ✓ No exponential blow up in the number of refutations
- ✓ Limits the side effects of PMRES
  - ✗ Several soft literals introduced for the same core may jointly belong to a new core
  - ✗ Processing order may affect subsequent cores
  - ✗ Subsequent cores usually larger than ONE

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

Handling weights

Soft literals in the unsatisfiable core

(a,5) (b,3) (c,2)

Handling weights

Soft literals in the unsatisfiable core

(a,5) (b,3) (c,2)

Replace it by

(a,2) (b,2) (c,2)
(a,3) (b,1)

Handling weights

Soft literals in the unsatisfiable core

(a,5) (b,3) (c,2)

Replace it by

(a,2) (b,2) (c,2)
(a,3) (b,1)

1. Increment lower bound by 2
2. Process soft literals in the first line
   - New soft literals of weight 2
3. Put back soft literals in the second line

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

Multi-level optimization (1)

Program Π

Π'		normalized	level 1	level 2	level 3
a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,1)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,1)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,1)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		

Multi-level optimization (1)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,1)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,1)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,1)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		
Π′		normalized	level 1	level 2	level 3

- Optimum stable models for the highest level can be extended to optimum stable models for Π

level	lb	soft literals	core	model	new constraints
3	0	*a*			

Multi-level optimization (1)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,1)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,1)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,1)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		
Π′		normalized	level 1	level 2	level 3

- Optimum stable models for the highest level can be extended to optimum stable models for Π

level	lb	soft literals	core	model	new constraints
3	0	a	—	a, d	a
2	0	b, d			

Multi-level optimization (1)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,1)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,1)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,1)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		
Π′		normalized	level 1	level 2	level 3

- Optimum stable models for the highest level can be extended to optimum stable models for Π

level	lb	soft literals	core	model	new constraints
3	0	a	—	a, d	a
2	0	b, d	b	—	
2	1	d			

Multi-level optimization (1)

Program Π

Π'		normalized	level 1	level 2	level 3
a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,1)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,1)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,1)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		

- Optimum stable models for the highest level can be extended to optimum stable models for Π

level	lb	soft literals	core	model	new constraints
3	0	a	—	a, d	a
2	0	b, d	b	—	
2	1	d	—	a, d	d
1	0	c, e			

Multi-level optimization (1)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,1)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,1)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,1)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		
Π′		normalized	level 1	level 2	level 3

- Optimum stable models for the highest level can be extended to optimum stable models for Π

level	lb	soft literals	core	model	new constraints
3	0	a	—	a, d	a
2	0	b, d	b	—	
2	1	d	—	a, d	d
1	0	c, e	c	—	
1	3	e			

Multi-level optimization (1)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,1)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,1)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,1)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		
Π′		normalized	level 1	level 2	level 3

- Optimum stable models for the highest level can be extended to optimum stable models for Π

level	lb	soft literals	core	model	new constraints
3	0	a	—	a, d	a
2	0	b, d	b	—	
2	1	d	—	a, d	d
1	0	c, e	c	—	
1	3	e	—	a, d, e	

Multi-level optimization (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,18)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,6)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,6)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		
Π′		normalized	level 1	level 2	level 3

Multi-level optimization (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,18)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,6)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,6)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		
Π′		normalized	level 1	level 2	level 3

- Level 1 has 3 sublevels: $\{e, c\}$, $\{d, b\}$, and $\{a\}$

level	lb	ub	soft lit.	core	model	new constraints
3	0	36	a			

Multi-level optimization (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,18)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,6)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,6)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		
Π′		normalized	level 1	level 2	level 3

- Level 1 has 3 sublevels: $\{e, c\}$, $\{d, b\}$, and $\{a\}$

level	lb	ub	soft lit.	core	model	new constraints
3	0	36	a	—	a, d	a
2	0	11	b, d			

Multi-level optimization (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,18)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,6)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,6)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		
Π′		normalized	level 1	level 2	level 3

- Level 1 has 3 sublevels: $\{e, c\}$, $\{d, b\}$, and $\{a\}$

level	lb	ub	soft lit.	core	model	new constraints
3	0	36	a	—	a, d	a
2	0	11	b, d	b	—	
2	6	11	d			

Multi-level optimization (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,18)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,6)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,6)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		
Π′		normalized	level 1	level 2	level 3

- Level 1 has 3 sublevels: $\{e, c\}$, $\{d, b\}$, and $\{a\}$

level	lb	ub	soft lit.	core	model	new constraints
3	0	36	a	—	a, d	a
2	0	11	b, d	b	—	
2	6	11	d	—	a, d	d
1	6	11	c, e			

Multi-level optimization (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,18)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,6)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,6)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		
Π′		normalized	level 1	level 2	level 3

- Level 1 has 3 sublevels: $\{e, c\}$, $\{d, b\}$, and $\{a\}$

level	lb	ub	soft lit.	core	model	new constraints
3	0	36	a	—	a, d	a
2	0	11	b, d	b	—	
2	6	11	d	—	a, d	d
1	6	11	c, e	c	—	
1	9	11	e			

Multi-level optimization (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,18)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,6)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,6)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		
Π′		normalized	level 1	level 2	level 3

- Level 1 has 3 sublevels: $\{e, c\}$, $\{d, b\}$, and $\{a\}$

level	lb	ub	soft lit.	core	model	new constraints
3	0	36	a	—	a, d	a
2	0	11	b, d	b	—	
2	6	11	d	—	a, d	d
1	6	11	c, e	c	—	
1	9	11	e	—	a, d, e	e
1	9	9	—			

Multi-level optimization (2)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [1@3]			(a,18)
b :– not not b.	:– a, c.	:∼ not b. [1@2]		(b,6)	
c :– not not c.	:– b, c.	:∼ not c. [3@1]	(c,3)		
d :– not not d.		:∼ not d. [1@2]		(d,6)	
e :– not not e.		:∼ not e. [2@1]	(e,2)		
Π′		normalized	level 1	level 2	level 3

- Level 1 has 3 sublevels: $\{e, c\}$, $\{d, b\}$, and $\{a\}$

	level	lb	ub	soft lit.	core	model	new constraints
	3	0	36	a	—	a, d	a
	2	0	11	b, d	b	—	
⇒	2	6	11	d	—	a, d	d
	1	6	11	c, e	c	—	
	1	9	11	e	—	a, d, e	e
	1	9	9	—			

Hardening: **lb** + weight of d $\geq$ **ub** $\Rightarrow$ d must be true to improve **ub**
Just add d as a constraint!

Outline: Part II

3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

Stratification (1)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [40@1]	(a,40)
b :– not not b.	:– a, c.	:∼ not b. [40@1]	(b,40)
c :– not not c.	:– b, c.	:∼ not c. [10@1]	(c,10)
d :– not not d.		:∼ not d. [20@1]	(d,20)
e :– not not e.		:∼ not e. [10@1]	(e,10)
Π′		normalized	soft literals

Stratification (1)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [40@1]	(a,40)
b :– not not b.	:– a, c.	:∼ not b. [40@1]	(b,40)
c :– not not c.	:– b, c.	:∼ not c. [10@1]	(c,10)
d :– not not d.		:∼ not d. [20@1]	(d,20)
e :– not not e.		:∼ not e. [10@1]	(e,10)
Π′		normalized	soft literals

Minimal unsatisfiable cores

1. (a,40) (b,40)
2. (a,40) (c,10)
3. (b,20) (c,10)

Stratification (1)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [40@1]	(a,40)
b :– not not b.	:– a, c.	:∼ not b. [40@1]	(b,40)
c :– not not c.	:– b, c.	:∼ not c. [10@1]	(c,10)
d :– not not d.		:∼ not d. [20@1]	(d,20)
e :– not not e.		:∼ not e. [10@1]	(e,10)
Π′		normalized	soft literals

Minimal unsatisfiable cores

1. (a,40) (b,40)
2. (a,40) (c,10)
3. (b,20) (c,10)

Lower bound increase

1. 40
2. 10
3. 10

Stratification (1)

Program Π

a :– not not a.	:– a, b.	:∼ not a. [40@1]	(a,40)
b :– not not b.	:– a, c.	:∼ not b. [40@1]	(b,40)
c :– not not c.	:– b, c.	:∼ not c. [10@1]	(c,10)
d :– not not d.		:∼ not d. [20@1]	(d,20)
e :– not not e.		:∼ not e. [10@1]	(e,10)
Π′		normalized	soft literals

Minimal unsatisfiable cores

1. (a,40) (b,40)
2. (a,40) (c,10)
3. (b,20) (c,10)

Lower bound increase

1. 40
2. 10
3. 10

Prefer unsatisfiable cores with largest minimum weight

Stratification (2)

1. Init stratum to the largest weight
2. Ignore soft literals whose weight is smaller than stratum
3. If there is an unsatisfiable core
   - Process it
   - Increment lower bound by stratum
   - Go to 2
4. Otherwise, a stable model is found
   - Possibly update upper bound, and cache the model
   - Decrease stratum
   - If stratum is 0 then go to 2
5. Optimum found

Outline: Part II

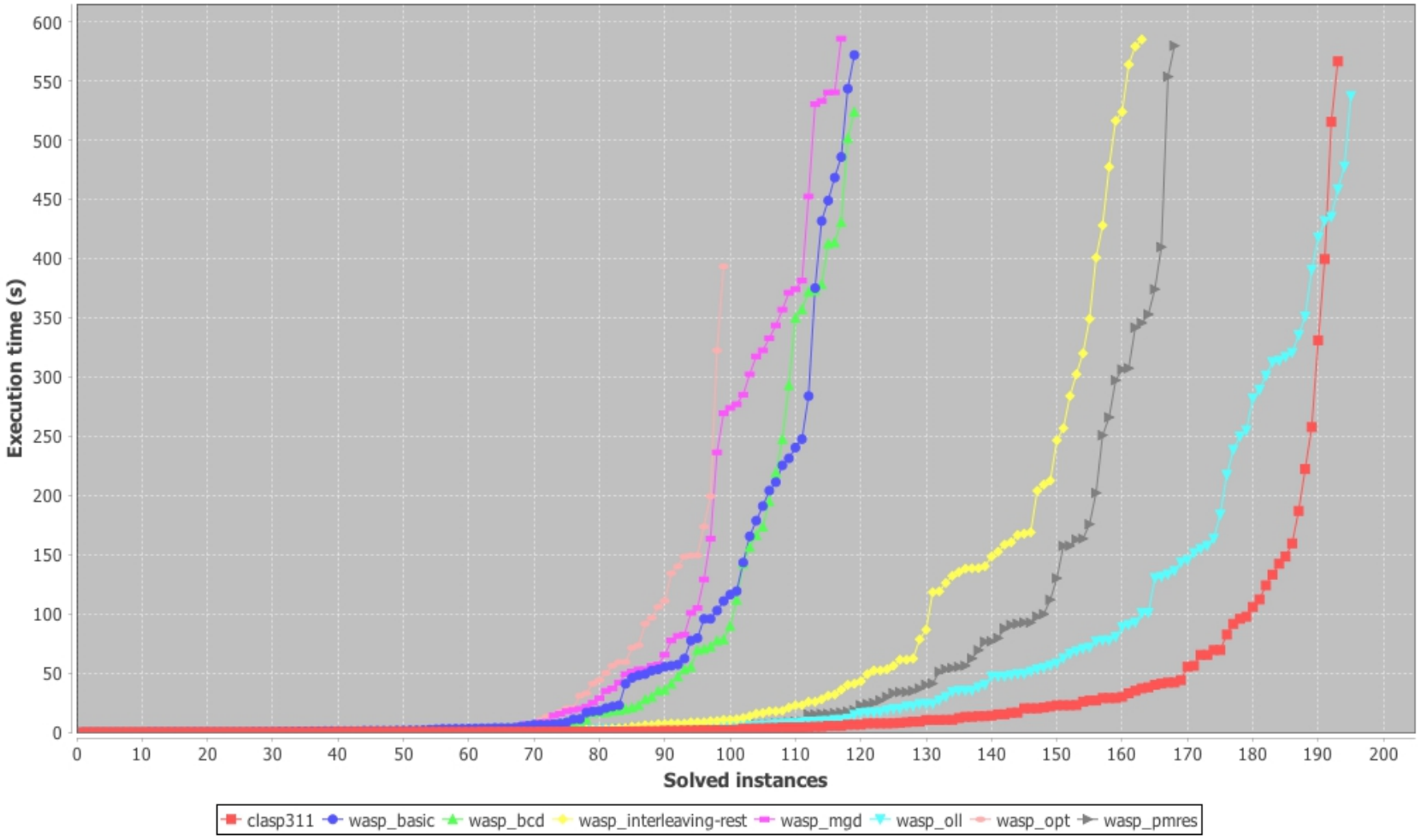
3 Optimization problems in ASP
- Branch and bound
- Binary search
- Core-based algorithms (overview)

4 Processing unsatisfiable cores
- OLL
- ONE
- PMRES
- K

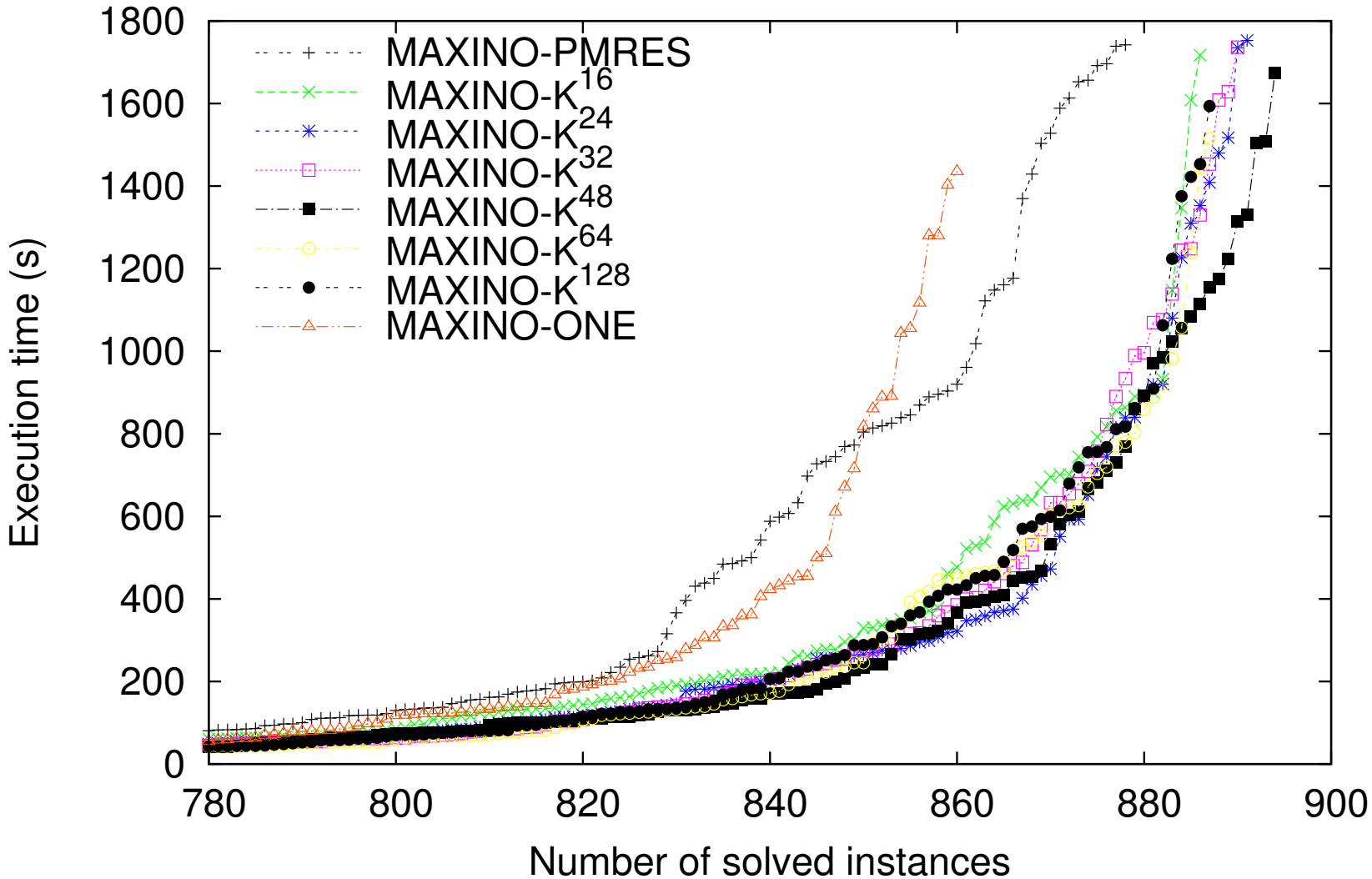
5 Weights and levels
- Handling weights
- Multi-level optimization
- Stratification

6 Experiment

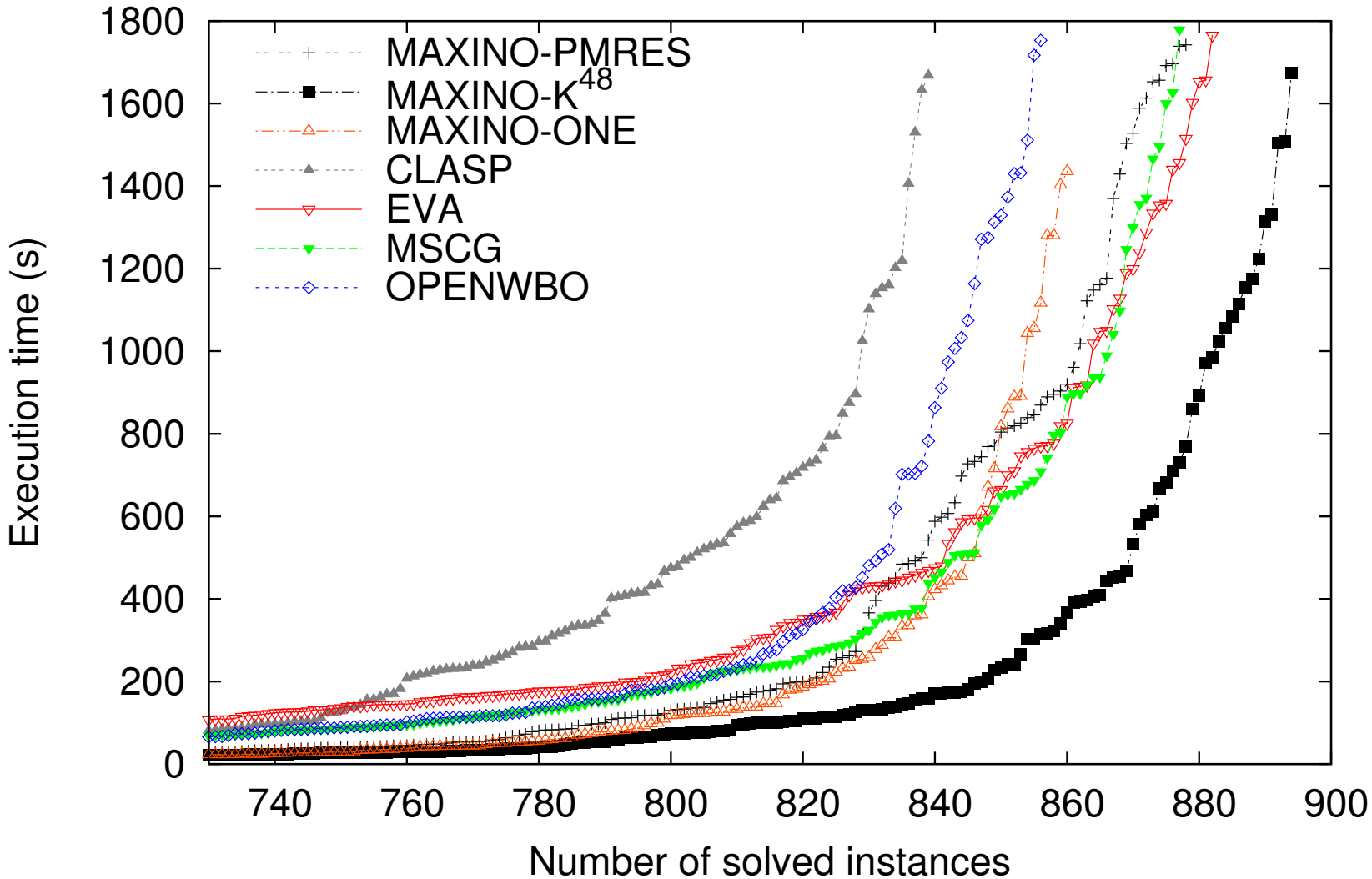
2014 ASP Competition: Overall



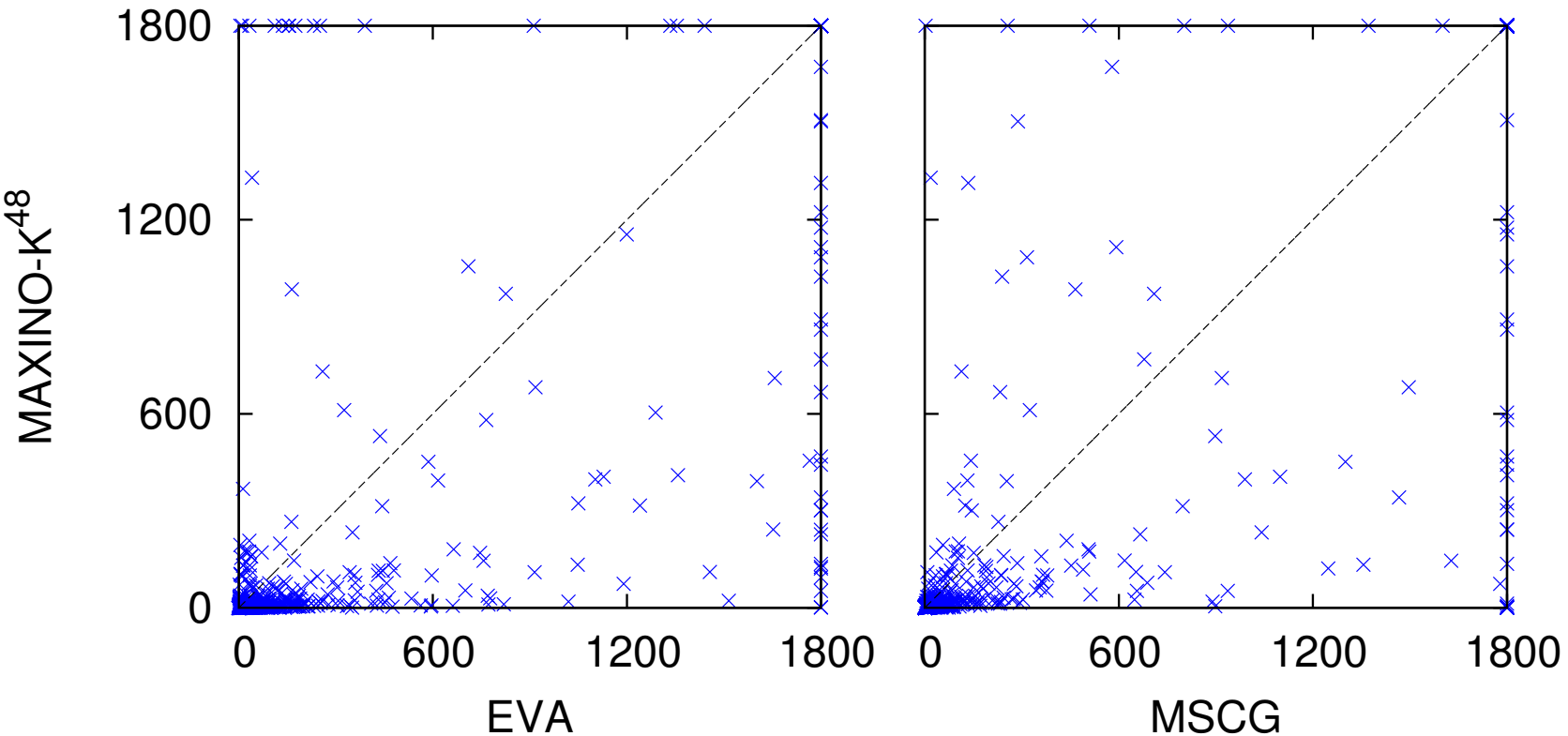
2014 MaxSAT Evaluation: Overall (1)



2014 MaxSAT Evaluation: Overall (2)



2014 MaxSAT Evaluation: Instance by instance



Conclusion

- ONE is similar to OLL, but much simpler

Conclusion

- ONE is similar to OLL, but much simpler
- K generalizes both ONE and PMRES

Conclusion

- ONE is similar to OLL, but much simpler
- K generalizes both ONE and PMRES
- Good performance with MAXINO
  `http://alviano.net/software/maxino/`

Conclusion

- ONE is similar to OLL, but much simpler
- K generalizes both ONE and PMRES
- Good performance with MAXINO
  `http://alviano.net/software/maxino/`
- MaxSAT algorithms and techniques extend to ASP

Conclusion

- ONE is similar to OLL, but much simpler
- K generalizes both ONE and PMRES
- Good performance with MAXINO
  `http://alviano.net/software/maxino/`
- MaxSAT algorithms and techniques extend to ASP
- Implemented by Carmine Dodaro in WASP
  `https://github.com/alviano/wasp`

Conclusion

- ONE is similar to OLL, but much simpler
- K generalizes both ONE and PMRES
- Good performance with MAXINO
  `http://alviano.net/software/maxino/`
- MaxSAT algorithms and techniques extend to ASP
- Implemented by Carmine Dodaro in WASP
  `https://github.com/alviano/wasp`

Thank you!