asprin: Answer Set Programming with Preferences

Javier Romero

University of Potsdam
Introduction

1. Introduction
2. Declarative Problem Solving
3. An Example
4. Preliminaries
5. Language
6. Implementation
7. Experimental analysis
   - Boosting optimization via heuristics
   - Dedicated systems versus asprin
8. Summary
Answer Set Programming (ASP) is an approach to declarative problem solving, combining

- a rich yet simple modeling language
- with high-performance solving capacities

tailored to Knowledge Representation and Reasoning

ASP allows for solving all search problems in \( NP \) (and \( NP^{NP} \)) in a uniform way

ASP is versatile as reflected by the ASP solver \( clingo \), winning first places at ASP, CASC, MISC, PB, and SAT competitions

ASP embraces many emerging application areas, and users
Answer Set Programming (ASP) is an approach to declarative problem solving, combining
- a rich yet simple modeling language
- with high-performance solving capacities
tailored to Knowledge Representation and Reasoning
ASP allows for solving all search problems in $NP$ (and $NP^{NP}$) in a uniform way
ASP is versatile as reflected by the ASP solver clingo, winning first places at ASP, CASC, MISC, PB, and SAT competitions
ASP embraces many emerging application areas, and users
Answer Set Programming (ASP) is an approach to declarative problem solving, combining a rich yet simple modeling language with high-performance solving capacities tailored to Knowledge Representation and Reasoning. ASP allows for solving all search problems in $NP$ (and $NP^{NP}$) in a uniform way. ASP is versatile as reflected by the ASP solver *clingo*, winning first places at ASP, CASC, MISC, PB, and SAT competitions. ASP embraces many emerging application areas, and users.
Answer Set Programming (ASP) is an approach to declarative problem solving, combining
- a rich yet simple modeling language
- with high-performance solving capacities
tailored to Knowledge Representation and Reasoning
ASP allows for solving all search problems in \( NP \) (and \( NP^{NP} \)) in a uniform way
ASP is versatile as reflected by the ASP solver \textit{clingo}, winning first places at ASP, CASC, MISC, PB, and SAT competitions
ASP embraces many emerging application areas, and users
For real-world applications, the identification of preferred, or optimal, solutions is often indispensable.

In many cases, this also involves the combination of various qualitative and quantitative preferences.

Only optimization statements representing objective functions using summation are established components of today’s ASP systems.

Example: \( \#\text{minimize}\{40 : \text{coast}; 70 : \text{mountain}\} \)
For real-world applications, the identification of preferred, or optimal, solutions is often indispensable. In many cases, this also involves the combination of various qualitative and quantitative preferences.

Only optimization statements representing objective functions using summation are established components of today’s ASP systems.

Example: `#minimize{40 : coast; 70 : mountain}`
asprin is a framework for handling preferences among the (stable) models of logic programs
- general because it captures many existing approaches to preference
- flexible because it allows combining different types of preferences
- extensible, allowing for an easy implementation of new approaches to preferences
**asprin** is a framework for handling preferences among the (stable) models of logic programs

- **general** because it captures many existing approaches to preference
- **flexible** because it allows combining different types of preferences
- **extensible**, allowing for an easy implementation of new approaches to preferences
asprin is a framework for handling preferences among the (stable) models of logic programs

- general because it captures many existing approaches to preference
- flexible because it allows combining different types of preferences
- extensible, allowing for an easy implementation of new approaches to preferences
asprin is a framework for handling preferences among the (stable) models of logic programs
  - general because it captures many existing approaches to preference
  - flexible because it allows combining different types of preferences
  - extensible, allowing for an easy implementation of new approaches to preferences
Declarative Problem Solving

Outline

1. Introduction
2. Declarative Problem Solving
3. An Example
4. Preliminaries
5. Language
6. Implementation
7. Experimental analysis
   - Boosting optimization via heuristics
   - Dedicated systems versus asprin
8. Summary
Informatics

“What is the problem?” versus “How to solve the problem?”
Informatics

“What is the problem?” versus “How to solve the problem?”

Problem

Computer

Solution

Output
Traditional programming

“What is the problem?” versus “How to solve the problem?”

Diagram:
- Problem
- Solution
- Computer
- Output
Traditional programming

“*What is the problem?*” versus “*How to solve the problem?*”

- Problem
- Solution
- Program
- Output

---

Programming

- Executing
- Interpreting

Javier Romero (KRR@UP)
asprin: ASP with Preferences
Declarative problem solving

“What is the problem?” versus “How to solve the problem?”

Problem

Solution

Computer

Output
Declarative Problem Solving

“What is the problem?” versus “How to solve the problem?”

Problem

Solution

Modeling

Interpreting

Representation

Output

Solving

Javier Romero (KRR@UP)
asprin: ASP with Preferences
ASP for Declarative problem solving

“What is the problem?” versus “How to solve the problem?”

Problem

Modeling

Logic Program

Solving

Solution

Interpreting

Stable Models
Outline

1. Introduction
2. Declarative Problem Solving
3. An Example
4. Preliminaries
5. Language
6. Implementation
7. Experimental analysis
   - Boosting optimization via heuristics
   - Dedicated systems versus asprin
8. Summary
An Example: Travelling Salesperson

Problem

Logic Program

Solving

Stable Models

Solution

Interpreting
Travelling Salesperson

- **Problem instance**  A map with cities, directed roads with distances, and one starting city
- **Problem class**  Find a route visiting every city once, and returning to the starting city

![Graph of a Travelling Salesperson problem instance with cities a, b, c, and d connected by directed roads with distances 3, 4, 5, and 5.](image)
An Example

Travelling Salesperson

- **Problem instance**  A map with cities, directed roads with distances, and one starting city
- **Problem class**  Find a route visiting every city once, and returning to the starting city

![Graph representation of the Travelling Salesperson problem instance.](image)
Travelling Salesperson

Problem

Logic Program

Solving

Stable Models

Solution

Interpreting

Modeling
Problem instance  A map with cities, directed roads with distances, and one starting city

- road(a,b,4). road(a,c,5). road(b,c,3). road(b,d,5).
- road(c,a,5). road(c,b,3). road(c,d,4).
- road(d,a,3). road(d,c,4).
Travelling Salesperson

- **Problem class**  Find a route visiting every city once, and returning to the starting city

```prolog
{ travel(X,Y) } :- road(X,Y,D).
visited(Y) :- start(X), travel(X,Y).
visited(Y) :- visited(X), travel(X,Y).
:- city(X), not visited(X).
:- city(X), 2 { travel(X,Y) }.
```
An Example

Travelling Salesperson

Problem class  Find a route visiting every city once, and returning to the starting city

\[
\{ \text{travel}(X,Y) \} :\text{- road}(X,Y,D).
\]

\[
\text{visited}(Y) :\text{- start}(X), \text{travel}(X,Y).
\]

\[
\text{visited}(Y) :\text{- visited}(X), \text{travel}(X,Y).
\]

\[
:\text{- city}(X), \text{not visited}(X).
\]

\[
:\text{- city}(X), 2 \{ \text{travel}(X,Y) \}.
\]
Travelling Salesperson

Problem class  Find a route visiting every city once, and returning to the starting city

\[
\{ \text{travel}(X,Y) \} :\!-\! \text{road}(X,Y,D).
\]

\[
\text{visited}(Y) :\!-\! \text{start}(X), \text{travel}(X,Y).
\]

\[
\text{visited}(Y) :\!-\! \text{visited}(X), \text{travel}(X,Y).
\]

\[
:\!-\! \text{city}(X), \text{not} \ \text{visited}(X).
\]

\[
:\!-\! \text{city}(X), 2 \ \{ \text{travel}(X,Y) \}.
\]
An Example

Travelling Salesperson

Problem

Modeling

Logic Program

Solving

Solution

Interpreting

Stable Models

Javier Romero (KRR@UP)
$ clingo tsp.lp 0

clingo version 5.1.0
Reading from tsp.lp
Solving...
Answer: 1
travel(a,b) travel(b,c) travel(c,d) travel(d,a) ...
Answer: 2
travel(a,b) travel(b,d) travel(d,c) travel(c,a) ...
Answer: 3
travel(a,c) travel(c,b) travel(b,d) travel(d,a) ...
SATISFIABLE

Models : 3
$ clingo tsp.lp 0

clingo version 5.1.0
Reading from tsp.lp
Solving...
Answer: 1
travel(a,b) travel(b,c) travel(c,d) travel(d,a) ...
Answer: 2
travel(a,b) travel(b,d) travel(d,c) travel(c,a) ...
Answer: 3
travel(a,c) travel(c,b) travel(b,d) travel(d,a) ...
SATISFIABLE

Models : 3
An Example

Travelling Salesperson

Problem

Modeling

Logic Program

Solving

Solution

Interpreting

Stable Models

asprin: ASP with Preferences
A route

Answer: 1

\(\text{travel}(a,b) \ \text{travel}(b,c) \ \text{travel}(c,d) \ \text{travel}(d,a) \ldots\)
Answer: 1
travel(a,b) travel(b,c) travel(c,d) travel(d,a) ...
A route

Answer: 2
travel(a,b) travel(b,d) travel(d,c) travel(c,a) ...
A route

Answer: 3

\text{travel(a,c) travel(c,b) travel(b,d) travel(d,a) ...}
An Example

**asprin** for Declarative Problem Solving

Problem

Modeling

Logic Program with Preferences

Solving

Optimal Stable Models

Solution

Interpreting
asprin for Declarative Problem Solving

**Problem**

Modeling

Logic Program with Preferences

Solving

Solution

Interpreting

Optimal Stable Models

asprin: ASP with Preferences
An Example

Travelling Salesperson

- Problem class: Find a route visiting every city once, and returning to the starting city. *Prefer routes of minimum distance.*

![Graph showing the Travelling Salesperson problem with cities a, b, c, and d connected by weighted edges.](image)
An Example

Travelling Salesperson

Problem

Logic Program with Preferences

Modeling

Solving

Solution

Optimal Stable Models

Interpreting
Travelling Salesperson

- Problem class: Find a route visiting every city once, and returning to the starting city. *Prefer routes of minimum distance.*

```plaintext
#preference(distance,less(weight)){
    D,(X,Y) :: travel(X,Y), road(X,Y,D)
}.
#optimize(distance).
```
Travelling Salesperson

Problem class Find a route visiting every city once, and returning to the starting city. Prefer routes of minimum distance.

```prolog
#preference(distance,less(weight)){
    D,(X,Y) :: travel(X,Y), road(X,Y,D)
}.
#optimize(distance).
```
### Travelling Salesperson

- **Problem class** Find a route visiting every city once, and returning to the starting city. *Prefer routes of minimum distance.*

```verbatim
#preference(distance, less(weight)) {
  D, (X,Y) :: travel(X,Y), road(X,Y,D)
}.

#optimize(distance).
```
Travelling Salesperson

- Problem
  - Logic Program with Preferences
- Solution
  - Optimal Stable Models

 Modeling - Interpreting

Solving
$ asprin tsp.lp preference.lp 0

asprin version 3.0.0
Reading from tsp.lp ...
Solving...
Answer: 1
  travel(a,c) travel(c,b) travel(b,d) travel(d,a) ...
Answer: 2
  travel(a,b) travel(b,c) travel(c,d) travel(d,a) ...
OPTIMUM FOUND

Models : 2
  Optimum : yes
  Optimal : 1
Travelling Salesperson

$ asprin tsp.lp preference.lp 0

asprin version 3.0.0
Reading from tsp.lp ...
Solving...
Answer: 1
travel(a,c) travel(c,b) travel(b,d) travel(d,a) ...
Answer: 2
travel(a,b) travel(b,c) travel(c,d) travel(d,a) ...
OPTIMUM FOUND

Models : 2
  Optimum : yes
  Optimal : 1
An Example

Travelling Salesperson

Problem

Modeling

Logic Program with Preferences

Solution

Interpreting

Optimal Stable Models

Solving
A route

Answer: 1
\text{travel}(a,c) \text{ travel}(c,b) \text{ travel}(b,d) \text{ travel}(d,a) \ldots

Answer: 2
\text{travel}(a,b) \text{ travel}(b,c) \text{ travel}(c,d) \text{ travel}(d,a) \ldots

OPTIMUM FOUND
An optimal route

Answer: 1
travel(a,c) travel(c,b) travel(b,d) travel(d,a) ...

Answer: 2
travel(a,b) travel(b,c) travel(c,d) travel(d,a) ...
OPTIMUM FOUND
An Example

Extending Travelling Salesperson

- **Problem class**  Find a route visiting every city once, and returning to the starting city. *Prefer going along the outside coast if it is cloudy, else prefer going through the inside mountains.*

```prolog
type(X,Y,mountain) :- road(X,Y,D), D == 5.
type(X,Y,coast) :- road(X,Y,D), D != 5.

#preference(weather,aso){
  travel(X,Y), type(X,Y,coast) >>
  travel(X',Y), type(X',Y,mountain) || cloudy;
  travel(X,Y), type(X,Y,mountain) >>
  travel(X',Y), type(X',Y,coast) || not cloudy
}.

#optimize(weather).
```

Javier Romero (KRR@UP)
Extending Travelling Salesperson

Problem class  Find a route visiting every city once, and returning to the starting city. *Prefer going along the outside coast if it is cloudy, else prefer going through the inside mountains.*

type(X,Y,mountain) :- road(X,Y,D), D == 5.
type(X,Y,coast) :- road(X,Y,D), D != 5.

#preference(weather,aso){
  travel(X,Y), type(X,Y,coast) >>
    travel(X’,Y), type(X’,Y,mountain) || cloudy;
  travel(X,Y), type(X,Y,mountain) >>
    travel(X’,Y), type(X’,Y,coast) || not cloudy
}.

#optimize(weather).
Extending Travelling Salesperson

Problem class Find a route visiting every city once, and returning to the starting city. *Prefer going along the outside coast if it is cloudy, else prefer going through the inside mountains.*

type(X,Y,mountain) :- road(X,Y,D), D == 5.
type(X,Y,coast) :- road(X,Y,D), D != 5.

#preference(weather,aso) {
    travel(X,Y), type(X,Y,coast) >>
    travel(X',Y), type(X',Y,mountain) || cloudy;
    travel(X,Y), type(X,Y,mountain) >>
    travel(X',Y), type(X',Y,coast) || not cloudy
}.

#optimize(weather).
An Example

Extending Travelling Salesperson

- Problem class  Find a route visiting every city once, and returning to the starting city. *Prefer going along the outside coast if it is cloudy, else prefer going through the inside mountains.*

```prolog
type(X,Y,mountain) :- road(X,Y,D), D == 5.
type(X,Y,coast) :- road(X,Y,D), D != 5.

#preference(weather,aso){
    travel(X,Y), type(X,Y,coast) >>
    travel(X',Y), type(X',Y,mountain) || cloudy;
    travel(X,Y), type(X,Y,mountain) >>
    travel(X',Y), type(X',Y,coast) || not cloudy
}.

#optimize(weather).
```

Javier Romero (KRR@UP)
### Extending Travelling Salesperson

```
$ asprin tsp.lp preference.lp 0

asprin version 3.0.0
Reading from tsp.lp ...
Solving...
Solving...
Answer: 1
travel(a,b) travel(b,c) travel(c,d) travel(d,a)
Answer: 2
travel(a,c) travel(b,d) travel(c,b) travel(d,a)
OPTIMUM FOUND
Answer: 3
travel(a,b) travel(b,d) travel(c,a) travel(d,c)
OPTIMUM FOUND

Models : 3
  Optimum : yes
  Optimal : 2
```
Extending Travelling Salesperson

$ asprin tsp.lp preference.lp 0

asprin version 3.0.0
Reading from tsp.lp ...
Solving...
Solving...
Answer: 1
travel(a,b) travel(b,c) travel(c,d) travel(d,a)
Answer: 2
travel(a,c) travel(b,d) travel(c,b) travel(d,a)
OPTIMUM FOUND
Answer: 3
travel(a,b) travel(b,d) travel(c,a) travel(d,c)
OPTIMUM FOUND

Models : 3
  Optimum : yes
  Optimal : 2
An Example

Extending Travelling Salesperson

- **Problem class** Find a route visiting every city once, and returning to the starting city. Prefer routes of minimum distance. Prefer going through the outside coast if it is cloudy, else prefer going through the inside mountains. *Combine both preferences with Pareto.*

```asprin
#preference(all,pareto){
/**distance;
/**weather
}
#optimize(all).
```
An Example

Extending Travelling Salesperson

- **Problem class** Find a route visiting every city once, and returning to the starting city. Prefer routes of minimum distance. Prefer going through the outside coast if it is cloudy, else prefer going through the inside mountains. *Combine both preferences with Pareto.*

```
#preference(all,pareto){
  **distance;
  **weather
}.
#optimize(all).
```
An Example

Extending Travelling Salesperson

$ asprin tsp.lp preference.lp 0

asprin version 3.0.0
Reading from tsp.lp ...
Solving...
Answer: 1
travel(a,b) travel(b,c) travel(c,d) travel(d,a)
OPTIMUM FOUND
Answer: 2
travel(a,c) travel(b,d) travel(c,b) travel(d,a)
OPTIMUM FOUND
Answer: 3
travel(a,b) travel(b,d) travel(c,a) travel(d,c)
OPTIMUM FOUND

Models : 3
  Optimum : yes
  Optimal : 3
Extending Travelling Salesperson

$ asprin tsp.lp preference.lp 0

asprin version 3.0.0
Reading from tsp.lp ...
Solving...
Answer: 1
travel(a,b) travel(b,c) travel(c,d) travel(d,a)
OPTIMUM FOUND
Answer: 2
travel(a,c) travel(b,d) travel(c,b) travel(d,a)
OPTIMUM FOUND
Answer: 3
travel(a,b) travel(b,d) travel(c,a) travel(d,c)
OPTIMUM FOUND

Models : 3
  Optimum : yes
  Optimal : 3
Outline

1 Introduction
2 Declarative Problem Solving
3 An Example
4 Preliminaries
5 Language
6 Implementation
7 Experimental analysis
   ▪ Boosting optimization via heuristics
   ▪ Dedicated systems versus asprin
8 Summary
A strict partial order $\succ$ on the stable models of a logic program. That is, $X \succ Y$ means that $X$ is preferred to $Y$.

A stable model $X$ is $\succ$-preferred, if there is no other stable model $Y$ such that $Y \succ X$.

A preference type is a (parametric) class of preference relations.

Example: subset, pareto, etc.
A strict partial order $\succ$ on the stable models of a logic program. That is, $X \succ Y$ means that $X$ is preferred to $Y$.

A stable model $X$ is $\succ$-preferred, if there is no other stable model $Y$ such that $Y \succ X$.

A preference type is a (parametric) class of preference relations.

Example: subset, pareto, etc.
A strict partial order $\succ$ on the stable models of a logic program.
That is, $X \succ Y$ means that $X$ is preferred to $Y$.

A stable model $X$ is $\succ$-preferred, if there is no other stable model $Y$ such that $Y \succ X$.

A preference type is a (parametric) class of preference relations.
Example: *subset*, *pareto*, etc.
naming atom

where \( s \) is the name of a preference

weighted formula

where each \( w_i \) is a term and \( \phi \) is a Boolean formula

preference element

where each \( \Phi_r \) is a set of weighted formulas

preference statement

where \( s \) and \( t \) represent the preference statement and its type and each \( e_j \) is a preference element

optimization directive

where \( s \) is the name of a preference

preference specification

is a set \( S \) of preference statements and a directive \( \#\text{optimize}(s) \) such that \( S \) is acyclic, closed, and \( s \in S \)
Language

- **naming atom**
  where $s$ is the name of a preference

- **weighted formula**
  where each $w_i$ is a term and $\phi$ is a Boolean formula

- **preference element**
  where each $\Phi_r$ is a set of weighted formulas

- **preference statement**
  where $s$ and $t$ represent the preference statement and its type and each $e_j$ is a preference element

- **optimization directive**
  where $s$ is the name of a preference

- **preference specification** is a set $S$ of preference statements and a directive $\#\text{optimize}(s)$ such that $S$ is acyclic, closed, and $s \in S$
Language

- naming atom
  - where $s$ is the name of a preference

- weighted formula
  - where each $w_i$ is a term and $\phi$ is a Boolean formula

- preference element
  - where each $\Phi_r$ is a set of weighted formulas

- preference statement
  - where $s$ and $t$ represent the preference statement and its type
    and each $e_j$ is a preference element

- optimization directive
  - where $s$ is the name of a preference

- preference specification
  - is a set $S$ of preference statements and a directive
    $\texttt{\#optimize}(s)$ such that $S$ is acyclic, closed, and $s \in S$
naming atom

where \( s \) is the name of a preference

weighted formula

where each \( w_i \) is a term and \( \phi \) is a Boolean formula

preference element

where each \( \Phi_r \) is a set of weighted formulas

preference statement

where \( s \) and \( t \) represent the preference statement and its type and each \( e_j \) is a preference element

optimization directive

where \( s \) is the name of a preference

preference specification is a set \( S \) of preference statements and a directive \( \#\text{optimize}(s) \) such that \( S \) is acyclic, closed, and \( s \in S \)
Language

- naming atom
  where \( s \) is the name of a preference

- weighted formula
  where each \( w_i \) is a term and \( \phi \) is a Boolean formula

- preference element
  where each \( \Phi_r \) is a set of weighted formulas

- preference statement
  where \( s \) and \( t \) represent the preference statement and its type
  and each \( e_j \) is a preference element

- optimization directive
  where \( s \) is the name of a preference

- preference specification
  is a set \( S \) of preference statements and a directive
  \#\ optimize\( (s) \) such that \( S \) is acyclic, closed, and \( s \in S \)
Language

- **naming atom**
  where \( s \) is the name of a preference

- **weighted formula**
  where each \( w_i \) is a term and \( \phi \) is a Boolean formula

- **preference element**
  where each \( \Phi_r \) is a set of weighted formulas

- **preference statement**
  where \( s \) and \( t \) represent the preference statement and its type and each \( e_j \) is a preference element

- **optimization directive**
  where \( s \) is the name of a preference

- **preference specification** is a set \( S \) of preference statements and a directive \#\text{optimize}(s)\) such that \( S \) is acyclic, closed, and \( s \in S \)
Language

- naming atom
  where \( s \) is the name of a preference

- weighted formula
  where each \( w_i \) is a term and \( \phi \) is a Boolean formula

- preference element
  where each \( \Phi_r \) is a set of weighted formulas

- preference statement
  where \( s \) and \( t \) represent the preference statement and its type
  and each \( e_j \) is a preference element

- optimization directive
  where \( s \) is the name of a preference

- preference specification is a set \( S \) of preference statements and a directive
  \( \#\text{optimize}(s) \) such that \( S \) is acyclic, closed, and \( s \in S \)
A preference type $t$ is a function mapping a set of preference elements $E$ to a preference relation

$$t(E) \subseteq \mathcal{A} \times \mathcal{A}$$

Examples

- $(X, Y) \in \text{subset}(E)$ iff $\{\ell \in E \mid X \models \ell\} \subset \{\ell \in E \mid Y \models \ell\}$
- $(X, Y) \in \text{pareto}(E)$ iff $\bigwedge_{s \in E} (X \preceq_s Y) \land \bigvee_{s \in E} (X \succeq_s Y)$
A preference type $t$ is a function mapping a set of preference elements $E$ to a preference relation

$$t(E) \subseteq A \times A$$

**Examples**

- $(X, Y) \in \text{subset}(E)$ iff $\{\ell \in E \mid X \models \ell\} \subseteq \{\ell \in E \mid Y \models \ell\}$
- $(X, Y) \in \text{pareto}(E)$ iff $\bigwedge_{\forall s \in E} (X \succeq_s Y) \land \bigvee_{\forall s \in E} (X \succ_s Y)$
A preference relation is obtained by applying a preference type to a set of preference elements.

\[ \text{#preference}(s, t) E \text{ declares preference relation } t(E), \text{ denoted by } \succ_s \]

Example \text{#preference}(1, \text{subset})\{a, b, c\} \text{ declares }

\[ X \succ_1 Y \text{ iff } \{ \ell \in \{a, b, c\} \mid X \models \ell \} \subseteq \{ \ell \in \{a, b, c\} \mid Y \models \ell \} \]
A preference relation is obtained by applying a preference type to a set of preference elements

\[ \text{preference}(s, t) E \] declares preference relation \( t(E) \), denoted by \( \succ_s \)

Example \[ \text{preference}(1, \text{subset}) \{a, b, c\} \] declares

\[ X \succ_1 Y \text{ iff } \{ \ell \in \{a, b, c\} \mid X \models \ell \} \subseteq \{ \ell \in \{a, b, c\} \mid Y \models \ell \} \]
A **preference relation** is obtained by applying a preference type to a set of preference elements

\[ \#\text{preference}(s, t) E \text{ declares preference relation } t(E), \text{ denoted by } \succ_s \]

**Example** \#\text{preference}(1, \text{subset})\{a, b, c\} declares

\[ X \succ_1 Y \text{ iff } \{ \ell \in \{a, b, c\} \mid X \models \ell \} \subset \{ \ell \in \{a, b, c\} \mid Y \models \ell \} \]
Outline

1. Introduction
2. Declarative Problem Solving
3. An Example
4. Preliminaries
5. Language
6. Implementation
7. Experimental analysis
   - Boosting optimization via heuristics
   - Dedicated systems versus asprin
8. Summary
Reification $H_X = \{\text{holds}(a) \mid a \in X\}$ and $H'_X = \{\text{holds}'(a) \mid a \in X\}$

Preference program Let $s$ be a preference statement declaring $\succ_s$. We define $Q_s$ as a preference program for $s$, if for all sets $X, Y \subseteq A$, we have

$$X \succ_s Y \iff Q_s \cup H_X \cup H'_Y$$

is satisfiable.

Note $Q_s$ is implemented as $F_s \cup E_{ts} \cup C$

Note *asprin*'s expressiveness is delineated by the decision problem.
Preference program

- Reification \( H_X = \{ \text{holds}(a) \mid a \in X \} \) and \( H'_X = \{ \text{holds}'(a) \mid a \in X \} \)

- Preference program Let \( s \) be a preference statement declaring \( \succsim_s \).
  We define \( Q_s \) as a preference program for \( s \), if for all sets \( X, Y \subseteq A \), we have
  \[
  X \succsim_s Y \iff Q_s \cup H_X \cup H'_Y \text{ is satisfiable}
  \]

- Note \( Q_s \) is implemented as \( F_s \cup E_{ts} \cup C \)

- Note \( asprin \)'s expressiveness is delineated by the decision problem
Preference program

- Reification \( H_X = \{ \text{holds}(a) \mid a \in X \} \) and \( H'_X = \{ \text{holds}'(a) \mid a \in X \} \)

- Preference program Let \( s \) be a preference statement declaring \( \succsim_s \).
  
  We define \( Q_s \) as a preference program for \( s \), if for all sets \( X, Y \subseteq A \), we have
  
  \[ X \succsim_s Y \text{ iff } Q_s \cup H_X \cup H'_Y \text{ is satisfiable} \]

- Note \( Q_s \) is implemented as \( F_s \cup E_{ts} \cup C \)

- Note \( asprin \)'s expressiveness is delineated by the decision problem
Preference program

- **Reification** $H_X = \{\text{holds}(a) \mid a \in X\}$ and $H'_X = \{\text{holds}'(a) \mid a \in X\}$
- **Preference program** Let $s$ be a preference statement declaring $\succ_s$. We define $Q_s$ as a preference program for $s$, if for all sets $X, Y \subseteq A$, we have
  \[ X \succ_s Y \iff Q_s \cup H_X \cup H'_Y \text{ is satisfiable} \]

- **Note** $Q_s$ is implemented as $F_s \cup E_{ts} \cup C$
- **Note** *asprin*'s expressiveness is delineated by the decision problem
Implementation

\[\# \text{preference}(1, \text{subset})\{a, b, c\}\]
\[\# \text{optimize}(1)\]

\[H\{a\} = \{ \text{holds}(a). \}\]

\[H'\{a,b\} = \{ \text{holds}'(a). \text{holds}'(b). \}\]

\[F_1 = \{ \text{preference}(1,\text{subset}). \text{preference}(1,1,1,\text{for}(a),()). \text{preference}(1,2,1,\text{for}(b),()). \text{preference}(1,3,1,\text{for}(c),()). \text{optimize}(1). \}\]

\[E_{\text{subset}} = \{ \text{better}(P) :- \text{preference}(P,\text{subset}), \text{not holds}(X), \text{holds}'(X), \text{preference}(P,_,_,\text{for}(X),_), \text{not holds}(Y) : \text{not holds}'(Y), \text{preference}(P,_,_,\text{for}(Y),_). \}\]

\[C = \{ :- \text{optimize}(P), \text{not better}(P). \}\]

There is a stable model, indicating that \(\{a\} \succ_1 \{a, b\}\).
\#preference(1, subset)\{a, b, c\} \\
\#optimize(1) \\

\[
H_{\{a\}} = \{ \text{holds}(a) \},
\]

\[
H'_{\{a, b\}} = \{ \text{holds'}(a), \text{holds'}(b) \},
\]

\[
F_1 = \{ \text{preference}(1, \text{subset})., \text{preference}(1, 1, 1, \text{for}(a), ())., \text{preference}(1, 2, 1, \text{for}(b), ())., \text{optimize}(1)., \text{preference}(1, 3, 1, \text{for}(c), ()). \}
\]

\[
E_{\text{subset}} = \{ \text{better}(P) :- \text{preference}(P, \text{subset}), \text{not holds}(X), \text{holds'}(X), \text{preference}(P, _, _, \text{for}(X), _), \text{not holds}(Y) : \text{not holds'}(Y), \text{preference}(P, _, _, \text{for}(Y), _). \}
\]

\[
C = \{ :- \text{optimize}(P), \text{not better}(P). \}
\]

There is a stable model, indicating that \(\{a\} \succ_1 \{a, b\}\)
There is a stable model, indicating that \( \{a\} \succ_1 \{a, b\} \).
\#preference(1, subset)\{a, b, c\} \\
\#optimize(1)

\[ H_{\{a,b\}} = \{ \text{holds}(a). \text{holds}(b). \} \]

\[ H'_{\{a\}} = \{ \text{holds}'(a). \} \]

\[ F_1 = \{ \text{preference}(1, \text{subset}). \text{pref}(1,1,1,\text{for}(a),()). \text{pref}(1,2,1,\text{for}(b),()). \text{pref}(1,3,1,\text{for}(c),()). \text{optimize}(1). \} \]

\[ E_{\text{subset}} = \{ \text{better}(P) :- \text{preference}(P, \text{subset}), \text{not} \text{holds}(X), \text{holds}'(X), \text{pref}(P,_,_,\text{for}(X),_), \text{not} \text{holds}(Y) : \text{not} \text{holds}'(Y), \text{pref}(P,_,_,\text{for}(Y),_). \} \]

\[ C = \{ :- \text{optimize}(P), \text{not} \text{better}(P). \} \]

There is no stable model, indicating that \{a, b\} \not\succ_1 \{a\}
Basic algorithm $solveOpt(P, s)$

**Input** : A program $P$ over $\mathcal{A}$ and preference statement $s$

**Output** : A $\succ_s$-preferred stable model of $P$, if $P$ is satisfiable, and $\bot$ otherwise

\[
Y \leftarrow solve(P) \\
\text{if } Y = \bot \text{ then return } \bot \\
\text{repeat} \\
\hspace{1em} X \leftarrow Y \\
\hspace{1em} Y \leftarrow solve(P \cup Q_s \cup R \cup H'_X) \\
\text{until } Y = \bot \\
\text{return } X
\]

where $R = \{holds(a) \leftarrow a \mid a \in \mathcal{A}\}$
asprin’s library

- Basic preference types
  - subset and superset
  - less(cardinality) and more(cardinality)
  - less(weight) and more(weight)
  - maxmin and minmax
  - aso (Answer Set Optimization)
  - poset (Qualitative Preferences)

- Composite preference types
  - neg
  - and
  - pareto
  - lexico

- And more to come...
  - cp (restricted CP nets)
asprin’s library

- Basic preference types
  - subset and superset
  - less(cardinality) and more(cardinality)
  - less(weight) and more(weight)
  - maxmin and minmax
  - aso (Answer Set Optimization)
  - poset (Qualitative Preferences)

- Composite preference types
  - neg
  - and
  - pareto
  - lexico

- And more to come...
  - cp (restricted CP nets)
Outline

1. Introduction
2. Declarative Problem Solving
3. An Example
4. Preliminaries
5. Language
6. Implementation
7. Experimental analysis
   - Boosting optimization via heuristics
   - Dedicated systems versus asprin
8. Summary
Outline

1. Introduction
2. Declarative Problem Solving
3. An Example
4. Preliminaries
5. Language
6. Implementation
7. Experimental analysis
   - Boosting optimization via heuristics
   - Dedicated systems versus asprin
8. Summary
Heuristics framework

- **clingo** allows for incorporating domain-specific heuristics into ASP solving
  - input language for expressing domain-specific heuristics
  - solving capacities for integrating domain-specific heuristics

- **Heuristic directive**  #heuristic

- **Heuristic modifiers** (atom, \(a\), and integer, \(v\))
  - **init** for initializing the heuristic value of \(a\) with \(v\)
  - **factor** for amplifying the heuristic value of \(a\) by factor \(v\)
  - **level** for ranking all atoms; the rank of \(a\) is \(v\)
  - **sign** for attributing the sign of \(v\) as truth value to \(a\)

- **Heuristic statements**

  #heuristic occurs(A,T) : action(A), time(T). [T,factor]
Heuristic framework

- **clingo** allows for incorporating domain-specific heuristics into ASP solving
  - input language for expressing domain-specific heuristics
  - solving capacities for integrating domain-specific heuristics

- **Heuristic directive**  
  - #heuristic

- **Heuristic modifiers**  
  - `init` for initializing the heuristic value of `a` with `v`
  - `factor` for amplifying the heuristic value of `a` by factor `v`
  - `level` for ranking all atoms; the rank of `a` is `v`
  - `sign` for attributing the sign of `v` as truth value to `a`

- **Heuristic statements**
  
  ```
  #heuristic occurs(A,T) : action(A), time(T). [T,factor]
  ```
Heuristic framework

- *clingo* allows for incorporating domain-specific heuristics into ASP solving
  - input language for expressing domain-specific heuristics
  - solving capacities for integrating domain-specific heuristics

- Heuristic directive  `#heuristic`

- Heuristic modifiers  \((\text{atom}, a, \text{and integer}, v)\)
  - `init` for initializing the heuristic value of \(a\) with \(v\)
  - `factor` for amplifying the heuristic value of \(a\) by factor \(v\)
  - `level` for ranking all atoms; the rank of \(a\) is \(v\)
  - `sign` for attributing the sign of \(v\) as truth value to \(a\)

- Heuristic statements

  `#heuristic occurs(A,T) : action(A), time(T). [T,factor]`
Heuristic framework

- **clingo** allows for incorporating domain-specific heuristics into ASP solving
  - input language for expressing domain-specific heuristics
  - solving capacities for integrating domain-specific heuristics

- Heuristic directive: `#heuristic`

- Heuristic modifiers (atom, \(a\), and integer, \(v\))
  - `init` for initializing the heuristic value of \(a\) with \(v\)
  - `factor` for amplifying the heuristic value of \(a\) by factor \(v\)
  - `level` for ranking all atoms; the rank of \(a\) is \(v\)
  - `sign` for attributing the sign of \(v\) as truth value to \(a\)

- Heuristic statements

  `#heuristic occurs(A,T) : action(A), time(T). [T,factor]`
Heuristic framework

- **clingo** allows for incorporating domain-specific heuristics into ASP solving
  - input language for expressing domain-specific heuristics
  - solving capacities for integrating domain-specific heuristics

- Heuristic directive `#heuristic`

- Heuristic modifiers
  - `init` for initializing the heuristic value of `a` with `v`
  - `factor` for amplifying the heuristic value of `a` by factor `v`
  - `level` for ranking all atoms; the rank of `a` is `v`
  - `sign` for attributing the sign of `v` as truth value to `a`

- Heuristic statements

  `#heuristic occurs(mv,5) : action(mv), time(5). [5,factor]`
Experimental analysis

Boosting optimization via heuristics

\textit{asprin} with different heuristic settings

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>System</th>
<th>\textit{asprin}_w</th>
<th>\textit{asprin}_w+s</th>
<th>\textit{asprin}_w+l</th>
<th>\textit{asprin}_w+f</th>
<th>\textit{asprin}_s</th>
<th>\textit{asprin}_s+s</th>
<th>\textit{asprin}_s+l</th>
<th>\textit{asprin}_s+f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ricochet</td>
<td>(30)</td>
<td>20.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timetabling</td>
<td>(12)</td>
<td>23687.75</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Puzzle</td>
<td>(7)</td>
<td>580.57</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crossing</td>
<td>(24)</td>
<td>211.92</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valves</td>
<td>(30)</td>
<td>56.63</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Expansion</td>
<td>(30)</td>
<td>7501.87</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Repair</td>
<td>(30)</td>
<td>6750.73</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diagnosis</td>
<td>(30)</td>
<td>1669.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\emptyset$</td>
<td></td>
<td>190 (129, 11)</td>
<td>130 (48, 6)</td>
<td>315 (6, 45)</td>
<td>217 (2, 31)</td>
<td>93 (105, 5)</td>
<td>72 (18, 10)</td>
<td>168 (3, 18)</td>
<td>101 (1, 21)</td>
</tr>
</tbody>
</table>

- \textit{asprin}_w — weight-based
- \textit{asprin}_w+s — sign heuristic
- \textit{asprin}_w+l — level-based heuristic
- \textit{asprin}_w+f — factor-based heuristic

- \textit{asprin}_s — subset-based
- \textit{asprin}_s+s — sign heuristic
- \textit{asprin}_s+l — level-based heuristic
- \textit{asprin}_s+f — factor-based heuristic
**asprin** with different heuristic settings

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>System</th>
<th>( \text{asprin}_w )</th>
<th>( \text{asprin}_w+s )</th>
<th>( \text{asprin}_w+l )</th>
<th>( \text{asprin}_w+f )</th>
<th>( \text{asprin}_s )</th>
<th>( \text{asprin}_s+s )</th>
<th>( \text{asprin}_s+l )</th>
<th>( \text{asprin}_s+f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ricochet</td>
<td>30</td>
<td>432 (8, 4)</td>
<td>407 (7, 4)</td>
<td>68 (1, 0)</td>
<td>71 (1, 0)</td>
<td>365 (8, 3)</td>
<td>461 (7, 10)</td>
<td>69 (1, 0)</td>
<td>71 (1, 0)</td>
</tr>
<tr>
<td>Timetabling</td>
<td>12</td>
<td>345 (285, 3)</td>
<td>255 (202, 2)</td>
<td>900 (4, 12)</td>
<td>416 (2, 1)</td>
<td>217 (144, 2)</td>
<td>21 (18, 0)</td>
<td>900 (2, 12)</td>
<td>5 (1, 0)</td>
</tr>
<tr>
<td>Puzzle</td>
<td>7</td>
<td>82 (2, 0)</td>
<td>112 (2, 0)</td>
<td>136 (2, 0)</td>
<td>805 (19, 20)</td>
<td>387 (6, 6)</td>
<td>32 (1, 0)</td>
<td>21 (1, 0)</td>
<td>51 (1, 0)</td>
</tr>
<tr>
<td>Crossing</td>
<td>24</td>
<td>104 (42, 1)</td>
<td>98 (35, 0)</td>
<td>460 (8, 11)</td>
<td>715 (0, 22)</td>
<td>38 (4, 0)</td>
<td>339 (4, 6)</td>
<td>673 (0, 21)</td>
<td>3 (1, 0)</td>
</tr>
<tr>
<td>Valves</td>
<td>30</td>
<td>69 (7, 0)</td>
<td>65 (6, 0)</td>
<td>38 (7, 0)</td>
<td>12 (3, 0)</td>
<td>38 (4, 0)</td>
<td>39 (4, 0)</td>
<td>3 (1, 0)</td>
<td>673 (0, 21)</td>
</tr>
<tr>
<td>Expansion</td>
<td>30</td>
<td>216 (299, 0)</td>
<td>10 (15, 0)</td>
<td>71 (3, 2)</td>
<td>8 (2, 0)</td>
<td>64 (295, 0)</td>
<td>14 (54, 0)</td>
<td>4 (4, 0)</td>
<td>3 (1, 0)</td>
</tr>
<tr>
<td>Repair</td>
<td>30</td>
<td>76 (48, 0)</td>
<td>15 (47, 0)</td>
<td>43 (4, 0)</td>
<td>118 (3, 2)</td>
<td>8 (43, 0)</td>
<td>3 (11, 0)</td>
<td>1 (1, 0)</td>
<td>1 (1, 0)</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>30</td>
<td>196 (341, 3)</td>
<td>76 (66, 0)</td>
<td>43 (4, 0)</td>
<td>217 (2, 31)</td>
<td>19 (338, 0)</td>
<td>2 (39, 0)</td>
<td>0 (1, 0)</td>
<td>0 (1, 0)</td>
</tr>
</tbody>
</table>

∅(∅, Σ)

- **asprin\(_w\)**
- **asprin\(_w+s\)**
- **asprin\(_w+l\)**
- **asprin\(_w+f\)**

- **w** — weight-based
- **s** — sign heuristic
- **l** — level-based heuristic
- **f** — factor-based heuristic

- **asprin\(_s\)**
- **asprin\(_s+s\)**
- **asprin\(_s+l\)**
- **asprin\(_s+f\)**

- **s** — subset-based
- **s** — sign heuristic
- **l** — level-based heuristic
- **f** — factor-based heuristic
### Experimental analysis

**Boosting optimization via heuristics**

**asprin** with different heuristic settings

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>System</th>
<th><strong>asprin</strong></th>
<th>asprin + s</th>
<th>asprin + l</th>
<th>asprin + f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ricochet</td>
<td>(30)</td>
<td>20.00</td>
<td>432 (8, 4)</td>
<td>68 (1, 0)</td>
<td>71 (1, 0)</td>
</tr>
<tr>
<td>Timetabling</td>
<td>(12)</td>
<td>23687.75</td>
<td>345 (285, 3)</td>
<td>900 (4, 12)</td>
<td>365 (8, 3)</td>
</tr>
<tr>
<td>Puzzle</td>
<td>(7)</td>
<td>580.57</td>
<td>82 (2, 0)</td>
<td>136 (2, 0)</td>
<td>416 (2, 1)</td>
</tr>
<tr>
<td>Crossing</td>
<td>(24)</td>
<td>211.92</td>
<td>104 (42, 1)</td>
<td>805 (19, 20)</td>
<td>387 (6, 6)</td>
</tr>
<tr>
<td>Valves</td>
<td>(30)</td>
<td>56.63</td>
<td>69 (7, 0)</td>
<td>460 (8, 11)</td>
<td>715 (0, 22)</td>
</tr>
<tr>
<td>Expansion</td>
<td>(30)</td>
<td>7501.87</td>
<td>216 (299, 0)</td>
<td>38 (7, 0)</td>
<td>12 (3, 0)</td>
</tr>
<tr>
<td>Repair</td>
<td>(30)</td>
<td>6750.73</td>
<td>76 (48, 0)</td>
<td>71 (3, 2)</td>
<td>8 (2, 0)</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>(30)</td>
<td>1669.00</td>
<td>196 (341, 3)</td>
<td>43 (4, 0)</td>
<td>118 (3, 2)</td>
</tr>
<tr>
<td>∅ (∅, Σ)</td>
<td></td>
<td>190 (129, 11)</td>
<td>315 (6, 45)</td>
<td>217 (2, 31)</td>
<td>93 (105, 5)</td>
</tr>
</tbody>
</table>

- asprin<sub>w</sub> — weight-based
- asprin<sub>s</sub> — subset-based
- asprin<sub>+</sub>s — sign heuristic
- asprin<sub>s</sub> — sign heuristic
- asprin<sub>+</sub>l — level-based heuristic
- asprin<sub>s</sub> — level-based heuristic
- asprin<sub>+</sub>f — factor-based heuristic
- asprin<sub>s</sub> — factor-based heuristic

---

**Javier Romero (KRR@UP)**

**asprin**: ASP with Preferences

48 / 55
Outline

1. Introduction
2. Declarative Problem Solving
3. An Example
4. Preliminaries
5. Language
6. Implementation
7. Experimental analysis
   - Boosting optimization via heuristics
   - Dedicated systems versus asprin
8. Summary
asprin versus clingo and metasp

(B,D,R&S; AAAI)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>System</th>
<th>clingo</th>
<th>asprin_w</th>
<th>asprin_w -f</th>
<th>metasp</th>
<th>asprin_s</th>
<th>asprin_s -f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ricochet</td>
<td>20.00</td>
<td>104.74 (0)</td>
<td>174.26 (0)</td>
<td>113.45 (0)</td>
<td>811.32 (24)</td>
<td>175.71 (0)</td>
<td>109.91 (0)</td>
</tr>
<tr>
<td>Timetabling</td>
<td>23687.75</td>
<td>35.82 (0)</td>
<td>490.39 (5)</td>
<td>694.92 (8)</td>
<td>798.75 (10)</td>
<td>142.03 (0)</td>
<td>12.01 (0)</td>
</tr>
<tr>
<td>Puzzle</td>
<td>580.57</td>
<td>77.00 (0)</td>
<td>77.39 (0)</td>
<td>96.70 (0)</td>
<td>34.79 (0)</td>
<td>17.06 (0)</td>
<td>17.22 (0)</td>
</tr>
<tr>
<td>Crossing</td>
<td>211.92</td>
<td>48.43 (0)</td>
<td>105.64 (1)</td>
<td>67.50 (0)</td>
<td>62.33 (0)</td>
<td>0.50 (0)</td>
<td>0.46 (0)</td>
</tr>
<tr>
<td>Valves</td>
<td>56.63</td>
<td>52.53 (0)</td>
<td>72.85 (0)</td>
<td>78.11 (0)</td>
<td>900.00 (30)</td>
<td>45.01 (0)</td>
<td>39.31 (0)</td>
</tr>
<tr>
<td>Expansion</td>
<td>7501.87</td>
<td>91.53 (0)</td>
<td>373.56 (2)</td>
<td>241.05 (7)</td>
<td>900.00 (30)</td>
<td>292.57 (0)</td>
<td>21.12 (0)</td>
</tr>
<tr>
<td>Repair</td>
<td>6750.73</td>
<td>71.78 (0)</td>
<td>102.19 (0)</td>
<td>43.94 (0)</td>
<td>900.00 (30)</td>
<td>6.88 (0)</td>
<td>2.19 (0)</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>1669.00</td>
<td>84.96 (0)</td>
<td>254.19 (3)</td>
<td>101.33 (0)</td>
<td>181.71 (6)</td>
<td>41.55 (0)</td>
<td>1.56 (0)</td>
</tr>
<tr>
<td>(\emptyset) ((\Sigma))</td>
<td></td>
<td>70.85 (0)</td>
<td>206.31 (11)</td>
<td>179.63 (15)</td>
<td>573.61 (130)</td>
<td>90.16 (0)</td>
<td>25.47 (0)</td>
</tr>
</tbody>
</table>

- **clingo** (using branch-and-bound)
- **asprin\_w**
- **asprin\_w -f**
  - \(w\) — weight-based
  - \(-f\) — no phase saving
- **metasp** (using disjunction)
- **asprin\_s**
- **asprin\_s -f**
  - \(s\) — subset-based
  - \(-f\) — no phase saving
asprin versus clingo and metasp  
(B,D,R&S; AAAI)

<table>
<thead>
<tr>
<th>Benchmark \ System</th>
<th>clingo</th>
<th>asprin w</th>
<th>asprin w -f</th>
<th>metasp</th>
<th>asprin s</th>
<th>asprin s -f</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ricochet</td>
<td>20.00</td>
<td>104.74 (0)</td>
<td>174.26 (0)</td>
<td>113.45 (0)</td>
<td>811.32 (24)</td>
<td>175.71 (0)</td>
</tr>
<tr>
<td>Timetabling</td>
<td>23687.75</td>
<td>35.82 (0)</td>
<td>490.39 (5)</td>
<td>694.92 (8)</td>
<td>798.75 (10)</td>
<td>142.03 (0)</td>
</tr>
<tr>
<td>Puzzle</td>
<td>580.57</td>
<td>77.00 (0)</td>
<td>77.39 (0)</td>
<td>96.70 (0)</td>
<td>34.79 (0)</td>
<td>17.06 (0)</td>
</tr>
<tr>
<td>Crossing</td>
<td>211.92</td>
<td>48.43 (0)</td>
<td>105.64 (1)</td>
<td>67.50 (0)</td>
<td>62.33 (0)</td>
<td>0.50 (0)</td>
</tr>
<tr>
<td>Valves</td>
<td>56.63</td>
<td>52.53 (0)</td>
<td>72.85 (0)</td>
<td>78.11 (0)</td>
<td>900.00 (30)</td>
<td>45.01 (0)</td>
</tr>
<tr>
<td>Expansion</td>
<td>7501.87</td>
<td>91.53 (0)</td>
<td>373.56 (2)</td>
<td>241.05 (7)</td>
<td>900.00 (30)</td>
<td>292.57 (0)</td>
</tr>
<tr>
<td>Repair</td>
<td>6750.73</td>
<td>71.78 (0)</td>
<td>102.19 (0)</td>
<td>43.94 (0)</td>
<td>900.00 (30)</td>
<td>6.88 (0)</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>1669.00</td>
<td>84.96 (0)</td>
<td>254.19 (3)</td>
<td>101.33 (0)</td>
<td>181.71 (6)</td>
<td>41.55 (0)</td>
</tr>
<tr>
<td>Ø(Σ)</td>
<td></td>
<td>70.85 (0)</td>
<td>206.31 (11)</td>
<td>179.63 (15)</td>
<td>573.61 (130)</td>
<td>90.16 (0)</td>
</tr>
</tbody>
</table>

- **clingo** (using branch-and-bound)
  - **asprin**
  - **asprin w**
  - **asprin w -f**

  - **w** — weight-based
  - **-f** — no phase saving

- **metasp** (using disjunction)
  - **asprin**
  - **asprin s**
  - **asprin s -f**

  - **s** — subset-based
  - **-f** — no phase saving
## aso versus asprin

<table>
<thead>
<tr>
<th></th>
<th>aso</th>
<th>aso&lt;br&gt;&lt;sub&gt;r&lt;/sub&gt;</th>
<th>asprin&lt;br&gt;&lt;sub&gt;a&lt;/sub&gt;</th>
<th>asprin&lt;br&gt;&lt;sub&gt;r+a&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>350</td>
<td>9 (0)</td>
<td>17 (0)</td>
<td>4 (0)</td>
<td>5 (0)</td>
</tr>
<tr>
<td>360</td>
<td>14 (0)</td>
<td>22 (0)</td>
<td>48 (0)</td>
<td>50 (0)</td>
</tr>
<tr>
<td>370</td>
<td>15 (0)</td>
<td>25 (0)</td>
<td>38 (0)</td>
<td>39 (0)</td>
</tr>
<tr>
<td>380</td>
<td>10 (0)</td>
<td>23 (0)</td>
<td>8 (0)</td>
<td>9 (0)</td>
</tr>
<tr>
<td>390</td>
<td>59 (0)</td>
<td>72 (0)</td>
<td>50 (1)</td>
<td>52 (1)</td>
</tr>
<tr>
<td>400</td>
<td>22 (0)</td>
<td>33 (0)</td>
<td>28 (0)</td>
<td>30 (0)</td>
</tr>
<tr>
<td>410</td>
<td>87 (1)</td>
<td>96 (1)</td>
<td>124 (2)</td>
<td>125 (2)</td>
</tr>
<tr>
<td>420</td>
<td>97 (1)</td>
<td>108 (1)</td>
<td>60 (0)</td>
<td>62 (0)</td>
</tr>
<tr>
<td>430</td>
<td>68 (0)</td>
<td>79 (0)</td>
<td>144 (0)</td>
<td>147 (0)</td>
</tr>
<tr>
<td>440</td>
<td>165 (3)</td>
<td>175 (3)</td>
<td>165 (2)</td>
<td>167 (2)</td>
</tr>
<tr>
<td>450</td>
<td>45 (0)</td>
<td>61 (0)</td>
<td>52 (0)</td>
<td>54 (0)</td>
</tr>
<tr>
<td>460</td>
<td>112 (1)</td>
<td>125 (1)</td>
<td>117 (2)</td>
<td>120 (2)</td>
</tr>
<tr>
<td>470</td>
<td>201 (4)</td>
<td>210 (4)</td>
<td>161 (2)</td>
<td>162 (2)</td>
</tr>
<tr>
<td>480</td>
<td>152 (2)</td>
<td>165 (2)</td>
<td>70 (1)</td>
<td>72 (1)</td>
</tr>
<tr>
<td>490</td>
<td>206 (2)</td>
<td>218 (2)</td>
<td>265 (4)</td>
<td>267 (4)</td>
</tr>
<tr>
<td>Φ (Σ)</td>
<td>84 (14)</td>
<td>95 (14)</td>
<td>89 (14)</td>
<td>91 (14)</td>
</tr>
</tbody>
</table>

- **aso** — dedicated system
- **aso<br><sub>r</sub>** — dedicated system with ranks
- **asprin<br><sub>a</sub>**
- **asprin<br><sub>r+a</sub>** — with ranks
## satpref versus asprin

<table>
<thead>
<tr>
<th>Benchmark \ System</th>
<th>satpref</th>
<th>satpref+s</th>
<th>satpref+H</th>
<th>asprin_0</th>
<th>asprin_0+s</th>
<th>asprin_0+H</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0 (29, 0)</td>
<td>0 (1, 0)</td>
<td>0 (1, 0)</td>
<td>1 (16, 0)</td>
<td>0 (2, 0)</td>
<td>0 (1, 0)</td>
</tr>
<tr>
<td>0.00621</td>
<td>0 (35, 0)</td>
<td>0 (1, 0)</td>
<td>90 (1, 6)</td>
<td>1 (17, 0)</td>
<td>1 (2, 0)</td>
<td>1 (1, 0)</td>
</tr>
<tr>
<td>0.01243</td>
<td>1 (75, 0)</td>
<td>1 (3, 0)</td>
<td>118 (1, 7)</td>
<td>6 (26, 0)</td>
<td>2 (3, 0)</td>
<td>3 (1, 0)</td>
</tr>
<tr>
<td>0.02486</td>
<td>8 (388, 0)</td>
<td>6 (10, 0)</td>
<td>635 (1, 38)</td>
<td>55 (74, 0)</td>
<td>9 (8, 0)</td>
<td>64 (1, 4)</td>
</tr>
<tr>
<td>0.04972</td>
<td>67 (1463, 2)</td>
<td>16 (36, 0)</td>
<td>900 (0.100)</td>
<td>318 (203, 16)</td>
<td>26 (17, 0)</td>
<td>176 (1.14)</td>
</tr>
<tr>
<td>1.0</td>
<td>850 (10315,88)</td>
<td>243 (590,10)</td>
<td>177 (1, 12)</td>
<td>856 (323, 92)</td>
<td>174 (96, 0)</td>
<td>280 (1,24)</td>
</tr>
<tr>
<td>$\emptyset(\emptyset, \Sigma)$</td>
<td>154 (2051,90)</td>
<td>44 (107,10)</td>
<td>320 (1,163)</td>
<td>206 (110,108)</td>
<td>35 (21, 0)</td>
<td>88 (1,42)</td>
</tr>
<tr>
<td>MAXSAT</td>
<td>54 (8849, 0)</td>
<td>9 (7, 0)</td>
<td>62 (1, 0)</td>
<td>835 (957, 31)</td>
<td>109 (31, 3)</td>
<td>171 (1, 6)</td>
</tr>
<tr>
<td>PBO/pbo-mqc-nencdr</td>
<td>5 (267, 0)</td>
<td>2 (2, 0)</td>
<td>664 (1, 88)</td>
<td>150 (207, 14)</td>
<td>9 (2, 0)</td>
<td>244 (1,20)</td>
</tr>
<tr>
<td>PBO/pbo-mqc-nlogencdr</td>
<td>3 (228, 0)</td>
<td>1 (2, 0)</td>
<td>237 (1, 21)</td>
<td>110 (214, 3)</td>
<td>5 (2, 0)</td>
<td>141 (1,15)</td>
</tr>
<tr>
<td>PSEUDO/primes</td>
<td>110 (396,18)</td>
<td>110 (1,18)</td>
<td>110 (1, 18)</td>
<td>215 (334, 27)</td>
<td>106 (5,17)</td>
<td>110 (1,17)</td>
</tr>
<tr>
<td>PSEUDO/routing</td>
<td>346 (409, 4)</td>
<td>49 (1, 0)</td>
<td>50 (1, 0)</td>
<td>85 (475, 0)</td>
<td>4 (1, 0)</td>
<td>86 (1, 1)</td>
</tr>
<tr>
<td>Partial-MINONE</td>
<td>14 (2, 0)</td>
<td>14 (2, 0)</td>
<td>7 (1, 0)</td>
<td>24 (2, 0)</td>
<td>24 (1, 0)</td>
<td>25 (1, 0)</td>
</tr>
<tr>
<td>$\emptyset(\emptyset, \Sigma)$</td>
<td>88 (1692,22)</td>
<td>31 (2,18)</td>
<td>188 (1,127)</td>
<td>236 (365, 75)</td>
<td>43 (7,20)</td>
<td>129 (1,59)</td>
</tr>
</tbody>
</table>

- **satpref**
- **satpref+s**
- **satpref+H**
- **s** — sign heuristic
- **H** — complex heuristic

- **asprin_0**
- **asprin_0+s**
- **asprin_0+H**
- **s** — sign heuristic
- **H** — complex heuristic

---

**Javier Romero**  
KRR@UP

**asprin**: ASP with Preferences

Page 53 / 55
Outline

1. Introduction
2. Declarative Problem Solving
3. An Example
4. Preliminaries
5. Language
6. Implementation
7. Experimental analysis
   - Boosting optimization via heuristics
   - Dedicated systems versus asprin
8. Summary
- **asprin** stands for “ASP for Preference handling”
- **asprin** is a general, flexible, and extendable framework for preference handling in ASP
- **asprin** caters to
  - off-the-shelf users using the preference relations in **asprin**’s library
  - preference engineers customizing their own preference relations
- **asprin** can be boosted by **clingo’**s heuristic framework
- [https://github.com/potassco/asprin](https://github.com/potassco/asprin)
asprin stands for “ASP for Preference handling”

asprin is a general, flexible, and extendable framework for preference handling in ASP

asprin caters to
  - off-the-shelf users using the preference relations in asprin’s library
  - preference engineers customizing their own preference relations

asprin can be boosted by clingo’s heuristic framework

https://github.com/potassco/asprin
asprin stands for “ASP for Preference handling”

asprin is a general, flexible, and extendable framework for preference handling in ASP

asprin caters to
- off-the-shelf users using the preference relations in asprin’s library
- preference engineers customizing their own preference relations

asprin can be boosted by clingo’s heuristic framework

https://github.com/potassco/asprin
asprin stands for “ASP for Preference handling”

asprin is a general, flexible, and extendable framework for preference handling in ASP

asprin caters to

- off-the-shelf users using the preference relations in asprin’s library
- preference engineers customizing their own preference relations

asprin can be boosted by clingo’s heuristic framework

https://github.com/potassco/asprin
asprin stands for “ASP for Preference handling”

asprin is a general, flexible, and extendable framework for preference handling in ASP

asprin caters to
- off-the-shelf users using the preference relations in asprin’s library
- preference engineers customizing their own preference relations

asprin can be boosted by clingo’s heuristic framework

https://github.com/potassco/asprin