ANSWER SET PROGRAMMING FOR JUDGMENT AGGREGATION

RONALD DE HAAN
ILLC, University of Amsterdam

MARIJA SLAVKOVIK
University of Bergen

Judgment aggregation

How to aggregate individual views on a set of logically related issues?

Judgment aggregation

user bought a toilet seat (p)

stop recommending toilet seats(q)

a user who bought a toilet was not recommended more toilet seats (r)

user was satisfied with service (t)

$$r \leftrightarrow (p \rightarrow q)$$ ← Constraint

AM Turk	$\{p$	q	r	$t\}$
W1	yes	no	no	no
W2	no	no	yes	yes
W3	yes	yes	yes	yes
Majority	yes	no	no	yes

Agenda: $\{p, q, r, t\}$

Profile: rows W1–W3

Collective judgment set / Outcome: Majority row

Aggregating judgment sets is hard

Judgment aggregation rule *F*	**Complexity of the outcome determination problem**
Condorcet rule CON	Σ_2^p-complete
Slater rule SLA	Θ_2^p-complete
Kemeny rule KEM	Θ_2^p-complete
MaxHamming rule MAXHAM	Θ_2^p-complete
AVGGEO-rule	Θ_3^p-complete
MAXGEO-rule	Θ_3^p-complete
Reversal-scoring rule REVSCO	Θ_2^p-complete
Ranked-agenda rule RAN	Σ_2^p-complete
LEXIMAX-rule	Δ_2^p-complete
Young rule YNG	Θ_2^p-complete
Dodgson rule DOD	Θ_2^p-complete

* Figure shamelessly borrowed from a manuscript in preparation that also includes Ulle Endriss and Jerome Lang

Answer set programming

Given is a declarative logic program,
a finite set of rules:

$$h_1 \vee \cdots \vee h_k \; :- \; b_1, \ldots, b_l, \text{not } b_{l+1}, \ldots, \text{not } b_t.$$

propositional literals

Answer set programming

Given is a declarative logic program,
a finite set of rules:

$$h_1 \vee \cdots \vee h_k \coloneq b_1, \ldots, b_l, \text{not } b_{l+1}, \ldots, \text{not } b_t.$$

propositional literals

Find a model (an answer set) that satisfies the rules.

Answer set programming

```
bird(tux). penguin(tux).
bird(tweety). chicken(tweety).
flies(X) :- bird(X), not -flies(X).
-flies(X) :- bird(X), not flies(X).
-flies(X) :- penguin(X).
```

Answer set 1: {bird(tux), penguin(tux), bird(tweety), chicken(tweety), flies(tweety), -flies(tux)}

Answer set 2: {bird(tux), penguin(tux), bird(tweety), chicken(tweety), -flies(tweety), -flies(tux)}

Find a model (an answer set) that satisfies the rules.

Judgment aggregation in ASP

$$r \leftrightarrow (p \rightarrow q)$$

AM Turk	p	q	r	t
W1	yes	no	no	no
W2	no	no	yes	yes
W3	yes	yes	yes	yes

```
issue(p).
issue (q).
issue(r).
issue(t).
```

Judgment aggregation in ASP

$$r \leftrightarrow (p \to q)$$

AM Turk	p	q	r	t
W1	yes	no	no	no
W2	no	no	yes	yes
W3	yes	yes	yes	yes

```
issue(p).   voter(1).
issue (q).  voter (2).
issue(r).   voter(3).
issue(t).
```

Judgment aggregation in ASP

$$r \leftrightarrow (p \rightarrow q)$$

AM Turk	p	q	r	t
W1	yes	no	no	no
W2	no	no	yes	yes
W3	yes	yes	yes	yes

```
issue(p).   voter(1).
issue (q).  voter (2).
issue(r).   voter(3).
issue(t).
```

Judgment aggregation in ASP

$$(\neg p \vee q \vee \neg r) \wedge (p \vee r) \wedge (\neg q \vee r)$$

AM Turk	p	q	r	t
W1	yes	no	no	no
W2	no	no	yes	yes
W3	yes	yes	yes	yes

```
issue(p).    voter(1).
issue (q).   voter (2).
issue(r).    voter(3).
issue(t).

clause(1,(-p;q;-r)).
```

Judgment aggregation in ASP

$$(\neg p \vee q \vee \neg r) \wedge (p \vee r) \wedge (\neg q \vee r)$$

AM Turk	p	q	r	t
W1	yes	no	no	no
W2	no	no	yes	yes
W3	yes	yes	yes	yes

```
issue(p).   voter(1).
issue (q).  voter (2).
issue(r).   voter(3).
issue(t).
clause(1,(-p;q;-r)).
clause(2,(p;r)).
```

Judgment aggregation in ASP

$$(\neg p \vee q \vee \neg r) \wedge (p \vee r) \wedge (\neg q \vee r)$$

AM Turk	p	q	r	t
W1	yes	no	no	no
W2	no	no	yes	yes
W3	yes	yes	yes	yes

```
issue(p).   voter(1).
issue (q).  voter (2).
issue(r).   voter(3).
issue(t).

clause(1,(-p;q;-r)).
clause(2,(p;r)).
clause(3,(-q;r)).
```

Judgment aggregation in ASP

$$(\neg p \vee q \vee \neg r) \wedge (p \vee r) \wedge (\neg q \vee r)$$

AM Turk	p	q	r	t
W1	yes	no	no	no
W2	no	no	yes	yes
W3	yes	yes	yes	yes

```
issue(p).   voter(1).
issue (q).  voter (2).
issue(r).   voter(3).
issue(t).

clause(1,(-p;q;-r)).
clause(2,(p;r)).
clause(3,(-q;r)).
```

```
js(1,p). js(1,-q). js(1,-r). js(1,-t).
```

Judgment aggregation in ASP

$$(\neg p \vee q \vee \neg r) \wedge (p \vee r) \wedge (\neg q \vee r)$$

AM Turk	p	q	r	t
W1	yes	no	no	no
W2	no	no	yes	yes
W3	yes	yes	yes	yes

```
issue(p).   voter(1).
issue (q).  voter (2).
issue(r).   voter(3).
issue(t).

clause(1,(-p;q;-r)).
clause(2,(p;r)).
clause(3,(-q;r)).

js(1,p). js(1,-q). js(1,-r). js(1,-t).
js(2,-p). js(2,-q). js(2,r). js(2,t).
```

Judgment aggregation in ASP

$$(\neg p \vee q \vee \neg r) \wedge (p \vee r) \wedge (\neg q \vee r)$$

AM Turk	p	q	r	t
W1	yes	no	no	no
W2	no	no	yes	yes
W3	yes	yes	yes	yes

```
issue(p).   voter(1).
issue (q).  voter (2).
issue(r).   voter(3).
issue(t).

clause(1,(-p;q;-r)).
clause(2,(p;r)).
clause(3,(-q;r)).

js(1,p). js(1,-q). js(1,-r). js(1,-t).
js(2,-p). js(2,-q). js(2,r). js(2,t).
js(3,p). js(3,q). js(3,r). js(3,t).
```

Judgment aggregation in ASP

$$(\neg p \vee q \vee \neg r) \wedge (p \vee r) \wedge (\neg q \vee r)$$

AM Turk	p	q	r	t
W1	yes	no	no	no
W2	no	no	yes	yes
W3	yes	yes	yes	yes

```
issue(p).   voter(1).
issue (q).  voter (2).
issue(r).   voter(3).
issue(t).

clause(1,(-p;q;-r)).
clause(2,(p;r)).
clause(3,(-q;r)).
```

```
js(1,p). js(1,-q). js(1,-r). js(1,-t).
js(2,-p). js(2,-q). js(2,r). js(2,t).
js(3,p). js(3,q). js(3,r). js(3,t).
```

```
agent(A) :- voter(A).
lit(X;-X) :- issue(X).
```

Judgment aggregation in ASP

$$(\neg p \vee q \vee \neg r) \wedge (p \vee r) \wedge (\neg q \vee r)$$

AM Turk	p	q	r	t
W1	yes	no	no	no
W2	no	no	yes	yes
W3	yes	yes	yes	yes

```
issue(p).    voter(1).
issue (q).   voter (2).
issue(r).    voter(3).
issue(t).
clause(1,(-p;q;-r)).
clause(2,(p;r)).
clause(3,(-q;r)).
```

```
js(1,p). js(1,-q). js(1,-r). js(1,-t).
js(2,-p). js(2,-q). js(2,r). js(2,t).
js(3,p). js(3,q). js(3,r). js(3,t).
```

```
agent(A) :- voter(A).
lit(X;-X) :- issue(X).
```

`1 { js(A,X) ; js(A,-X) } 1 :- agent(A), issue(X).`

enforcers completeness

Judgment aggregation in ASP

$$(\neg p \vee q \vee \neg r) \wedge (p \vee r) \wedge (\neg q \vee r)$$

AM Turk	p	q	r	t
W1	yes	no	no	no
W2	no	no	yes	yes
W3	yes	yes	yes	yes

```
issue(p).   voter(1).
issue (q).  voter (2).
issue(r).   voter(3).
issue(t).
clause(1,(-p;q;-r)).
clause(2,(p;r)).
clause(3,(-q;r)).
```

```
js(1,p). js(1,-q). js(1,-r). js(1,-t).
js(2,-p). js(2,-q). js(2,r). js(2,t).
js(3,p). js(3,q). js(3,r). js(3,t).
```

```
agent(A) :- voter(A).
lit(X;-X) :- issue(X).
1 { js(A,X) ; js(A,-X) } 1 :- agent(A), issue(X)
:- agent(A), clause(C,_), js(A,-L) : clause(C,L).
```

enforcers consistency with constraint

We implemented

Judgment aggregation rule F	**Complexity of the outcome determination problem**	
Condorcet rule CON	Σ_2^p-complete	(Theorems 4.1 and 4.2)
Slater rule SLA	Θ_2^p-complete	(Theorems 4.3 and 4.4)
Kemeny rule KEM	Θ_2^p-complete	(Theorems 4.5 and 4.6)
MaxHamming rule MAXHAM	Θ_2^p-complete	(Theorems 4.7 and 4.8)
Reversal-scoring rule REVSCO	Θ_2^p-complete	(Theorems 4.14 and 4.15)
Ranked-agenda rule RAN	Σ_2^p-complete	(Theorems 4.16 and 4.17)
LEXIMAX-rule	Δ_2^p-complete	(Theorems 4.18 and 4.19)
Young rule YNG	Θ_2^p-complete	(Theorems 4.20 and 4.21)
Dodgson rule DOD	Θ_2^p-complete	(Theorems 4.22 and 4.23)

Majority

```
issue(p).   voter(1).   clause(1,(-p;q;-r)). js(1,p). js(1,-q). js(1,-r). js(1,-t).
issue (q).  voter (2).  clause(2,(p;r)).     js(2,-p). js(2,-q). js(2,r). js(2,t).
issue(r).   voter(3).   clause(3,(-q;r)).    js(3,p). js(3,q). js(3,r). js(3,t).
issue(t).
agent(A) :- voter(A).
lit(X;-X) :- issue(X).
1 { js(A,X) ; js(A,-X) } 1 :- agent(A), issue(X)
:- agent(A), clause(C,_), js(A,-L) : clause(C,L).
```

```
pc(X,N) :- lit(X), N = #count { A : voter(A), js(A,X) }.
maj(X) :- lit(X), pc(X,N), pc(-X,M), N > M.
js(col,X) :- maj(X).
agent(col).
```

What we did

Judgment aggregation aggregation rules

Preference aggregation rules 

Agenda properties

Profile properties

GitHub, Inc. [US] | https://github.com/rdehaan/ja-asp

GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

Sign up

Encodings of Judgment Aggregation (JA) problems into Answer Set Programming (ASP)

70 commits | 1 branch | 0 releases | 1 contributor | GPL-3.0

Branch: master | New pull request | Find File | Clone or download

rdehaan Update README — Latest commit 6f107a3 on 29 Sep 2018

agenda-properties	Rename 'single-crossedness' to 'unidimensional alignment'	11 months ago
examples	Fix examples/profile/profile3.lp	11 months ago
profile-properties	Rename 'single-crossedness' to 'unidimensional alignment'	11 months ago
windet	Add (naive) encoding of Kemeny based on saturation	11 months ago
.gitignore	Create todos.md	last year
LICENSE	Initial commit	last year
README.md	Update README	11 months ago
ja.lp	Encode the graph of all complete & consistent judgment sets	11 months ago
meta.lp	Import files	last year
metaD.lp	Improve file organization	last year
metaO.lp	Improve file organization	last year
models.lp	Add support for auxiliary variables in the encodings	last year
pretty-printing.lp	Rename 'single-crossedness' to 'unidimensional alignment'	11 months ago
todos.md	Update todos	11 months ago

Thank you