Aggregating Lexicographic Preferences Over Combinatorial Domains

Xudong Liu

Department of Computer Science
University of Kentucky
Lexington, KY, USA

November 6, 2013

Multi-Agent Decision Making

A group of students at UK are deciding on a joint vacation in 2014 and they may consider the following issues:

- Time
- Destination
- Transportation

(a) Domain of Time

(b) Preference

Figure: Time

Multi-Agent Decision Making

(a) Domain of Destination

(b) Preference

Figure: Destination

Multi-Agent Decision Making

(a) Domain of Transportation

(b) Preference

Figure: Transportation

Multi-Agent Decision Making

Aggregating agents' individual preferences over alternatives to achieve collaborative decisions.

1. Alternatives: *combinatorial* domains
2. Individual preferences: *lexicographic preference trees (LP trees)*
3. Aggregation: based on *positional scoring voting rules*
4. Collaborative decisions: alternatives with the highest score

1. Issues: $X = \{X_1, X_2, \ldots, X_p\}$, with $D(X_i) = \{0_i, 1_i\}$
2. Alternatives: $\mathcal{X} = D(X_1) \times D(X_2) \times \ldots \times D(X_p)$, $|\mathcal{X}| = 2^p$ (denoted by m)
3. Vote: a strict total order over $\mathcal{X}$
4. Profile: a finite set of votes collected from n voters

Figure: Alternatives in the vacation planning problem

Positional Scoring Rules

1. Scoring vector: $w = (w_1, \ldots, w_m)$ of non-negative integers , where $w_1 \geq w_2 \geq \ldots \geq w_m$ and $w_1 > w_m$.
2. Let $v = o_1 \succ o_2 \succ \ldots \succ o_m$ be a vote over $\mathcal{X}$, the score of alternative o_i in vote v is w_i, denoted by $s_w(o_i, v)$
3. Let P be a profile of votes, the score of alternative o_i in profile P:

$$s_w(o_i, P) = \sum_{v \in P} s_w(o_i, v)$$

Positional Scoring Rules

- k-approval: $(1, \ldots, 1, 0, \ldots, 0)$ with k being the number of 1's and $m - k$ the number of 0's where $m = 2^p$
- Borda: $(m-1, m-2, \ldots, 1, 0)$
- (k, l)-approval: $(a, \ldots, a, b, \ldots, b, 0 \ldots, 0)$, where a and b are constants ($a > b$) and the numbers of a's and b's equal to k and l, respectively

1. An *LP tree* [1] $\mathcal{L}$ over $\mathcal{I} = \{X_1, \ldots, X_p\}$ is a *binary tree*
2. Each node t in $\mathcal{L}$ is labeled by an issue from $\mathcal{I}$ and with *preference information* $(0 > 1$ or $1 > 0)$
3. Each issue appears **exactly once** on each path from the root to a leaf

[1] Booth, Chevaleyre, Lang, Mengin, and Sombattheera. *Learning conditionally lexicographic preference relations*, 2010.

LP Trees: Example

$$D(Time) = \{summer(s), winter(w)\},\ D(Dest) = \{Chicago(c), Miami(m)\},$$
$$D(Tran) = \{drive(d), fly(f)\}$$

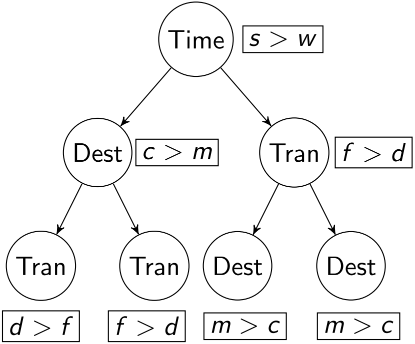


Figure: LP tree

LP Trees: Example

$$D(Time) = \{summer(s), winter(w)\},\ D(Dest) = \{Chicago(c), Miami(m)\},$$
$$D(Tran) = \{drive(d), fly(f)\}$$

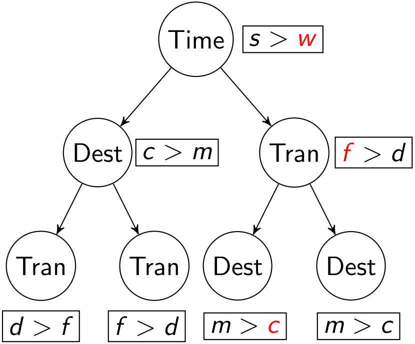
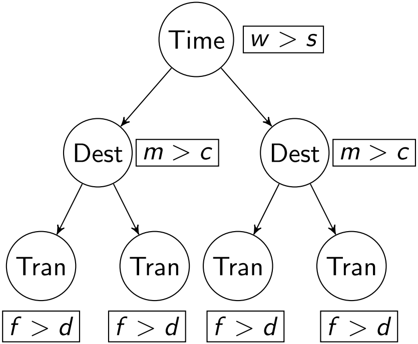


Figure: LP tree

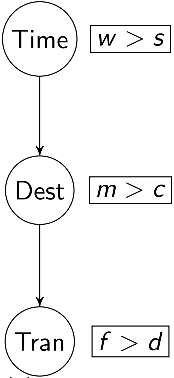
$$scd \succ scf \succ smf \succ smd \succ wmf \succ wcf \succ wmd \succ wcd$$

Collapsed LP Trees

$$D(Time) = \{summer(s), winter(w)\},\ D(Dest) = \{Chicago(c), Miami(m)\},$$
$$D(Tran) = \{drive(d), fly(f)\}$$



(a) Full LP tree



(b) Collapsed LP tree

Figure: Collapse to UI-UP

Collapsed LP Trees

$$D(Time) = \{summer(s), winter(w)\},\ D(Dest) = \{Chicago(c), Miami(m)\},$$
$$D(Tran) = \{drive(d), fly(f)\}$$

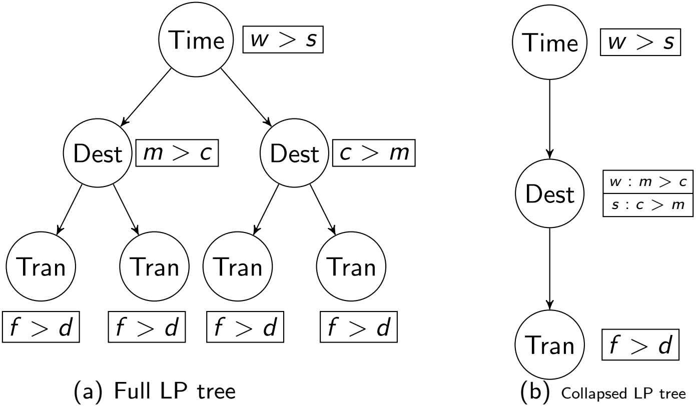
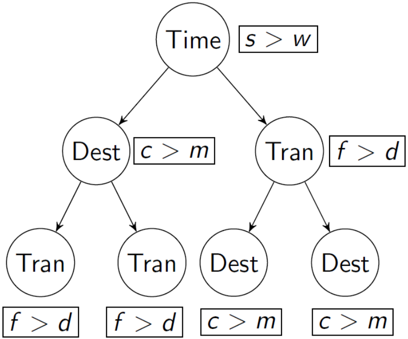
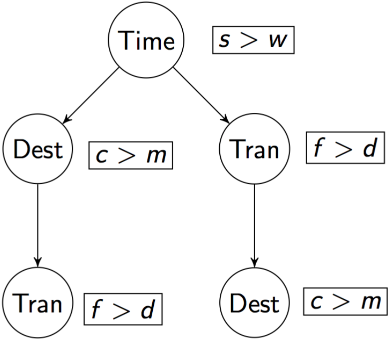


Figure: Collapse to UI-CP

$$D(Time) = \{summer(s), winter(w)\},\ D(Dest) = \{Chicago(c), Miami(m)\},$$
$$D(Tran) = \{drive(d), fly(f)\}$$



(a) Full LP tree

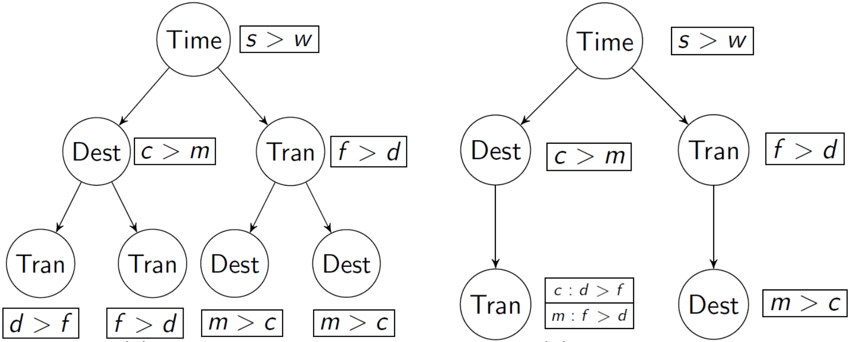


(b) Collapsed LP tree

Figure: Collapse to CI-UP

Collapsed LP Trees

$$D(Time) = \{summer(s), winter(w)\},\ D(Dest) = \{Chicago(c), Miami(m)\},$$
$$D(Tran) = \{drive(d), fly(f)\}$$



(a) Full LP tree (b) Collapsed LP tree

Figure: Collapse to CI-CP

The Evaluation and Winner Problems

The Evaluation Problem

Let r be a positional scoring rule with a scoring vector w, $\mathcal{C}$ a class of LP trees. Given a $\mathcal{C}$-profile P of n LP trees over p issues and a positive integer R, the *evaluation* problem is to decide whether there exists an alternative $o \in \mathcal{X}$ such that $s_w(o, P) \geq R$.

The Winner Problem

Let r be a positional scoring rule with a scoring vector w, $\mathcal{C}$ a class of LP trees. Given a $\mathcal{C}$-profile P of n LP trees over p issues, the *winner* problem is to compute an alternative $o \in \mathcal{X}$ with the maximum score $s_w(o, P)$.

Computational Complexity

- The computational complexity of the winner and the evaluation problems has *not* been fully understood
- With known results for some special cases, we aim at expanding the space of known results for *more general* positional scoring rules
- All computational complexity results and algorithms assume *compact* representations of LP trees

Computational Complexity: k-Approval

1. If k is fixed or $k = 2^{p-1} \pm f(p)$, where $f(p)$ is a polynomial of p, we have

	UP	CP
UI	P	P
CI	P	P

(a) Evaluation

	UP	CP
UI	P	P
CI	P	P

(b) Winner

Figure: k-approval [2]

[2] The case where $f(p) = 0$ is shown by Lang, Mengin, and Xia. *Aggregating conditionally lexicographic preferences on multi-issue domains*, 2012. Other cases are new results we obtained.

1. if $k = c \cdot 2^{p-M}$ (c and M are constants) and $k \neq 2^{p-1}$, we have

	UP	CP
UI	NPC	NPC
CI	NPC	NPC

(a) Evaluation

	UP	CP
UI	NP-Hard	NP-Hard
CI	NP-Hard	NP-Hard

(b) Winner

Figure: k-approval [3]

[3] Lang, Mengin, and Xia. *Aggregating conditionally lexicographic preferences on multi-issue domains*, 2012.

Computational Complexity: Borda

	UP	CP
UI	P	NPC
CI	NPC	NPC

(a) Evaluation

	UP	CP
UI	P	NP-Hard
CI	NP-Hard	NP-Hard

(b) Winner

Figure: Borda [4]

[4] Lang, Mengin, and Xia. *Aggregating conditionally lexicographic preferences on multi-issue domains*, 2012.

Our research considers yet another class of positional scoring rules: (k, l)-approval

$(2^{p-2}, 2^{p-2})$-Approval Evaluation Problem

Let w be the scoring vector for $(2^{p-2}, 2^{p-2})$-approval. The problem to decide for a given *UI-UP* profile P and a given positive integer R whether there is an alternative o such that $s_w(P, o) \geq R$ is NP-complete.

Proof.

Hardness follows from a polynomial reduction from the NP-complete problem *MIN 2-SAT* [5].

[5] Given a set Φ of 2-clauses and a positive integer l, the problem is to decide if there is an assignment satisfying at most l clauses in Φ.

If $k = l = 2^{p-2}$ or $k = l = 2^{p-3}$, we have

	UP	CP
UI	NPC	NPC
CI	NPC	NPC

(a) Evaluation

	UP	CP
UI	NP-Hard	NP-Hard
CI	NP-Hard	NP-Hard

(b) Winner

Figure: (k, l)-approval [6]

[6] Liu and Truszczynski, *Aggregating Conditionally Lexicographic Preferences Using Answer Set Programming*, ADT, 2013.

Computational Techniques

- Computational methods need to be implemented and tested to see how problems with different settings of parameters could be handled
- ASP and W-MAXSAT tools are chosen because they are designed to address NP-hard problems
- Our goal is to understand the scope of applicability of the solvers and compare their effectiveness

Modeling the Problems in ASP

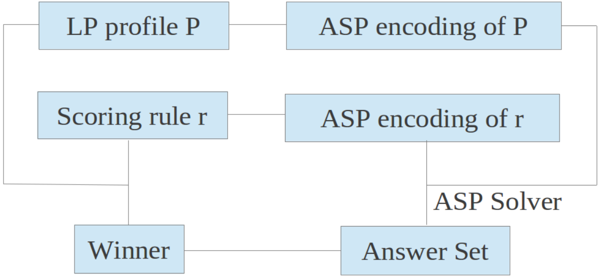


Figure: The winner problem

- Solvers: *clingo*, *clingcon*

Modeling the Problems in ASP

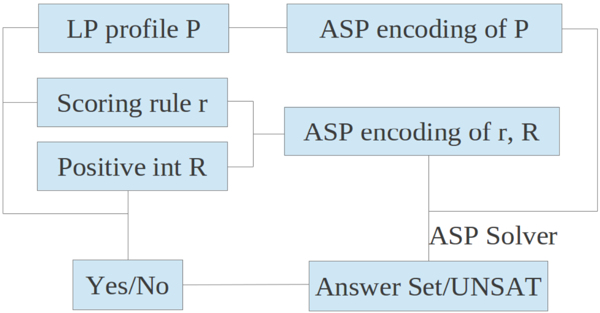


Figure: The evaluation problem

- Solvers: *clingo*, *clingcon*

Modeling the Winner Problem as a W-MAXSAT Problem

Weighted Maximum Satisfiability Problem (W-MAXSAT)

Let X be a set of boolean variables $\{X_1, \ldots, X_p\}$, Φ a set of weighted clauses $\{c_1 : w_1, \ldots, c_n : w_n\}$ over X, where each w_i is a positive integer, the problem is to find a truth assignment over X to maximize the sum of weights of satisfied clauses in Φ.

- Solver: *toulbar*

Modeling the Winner Problem as a W-MAXSAT Problem

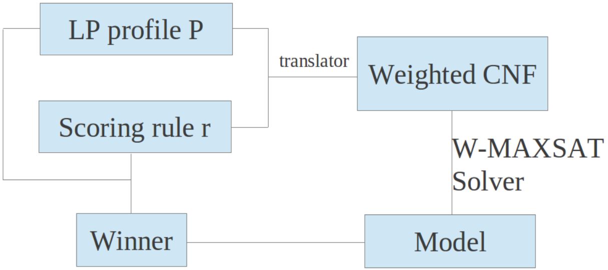
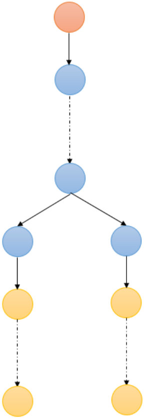


Figure: The winner problem

Random LP Profiles

- To experiment with LP profiles, we developed methods to randomly generate *encodings* of a special type of CI-CP LP tree of size linear in the number of issues



0_i	$0_r > 1_r$
1_i	$1_r > 0_r$

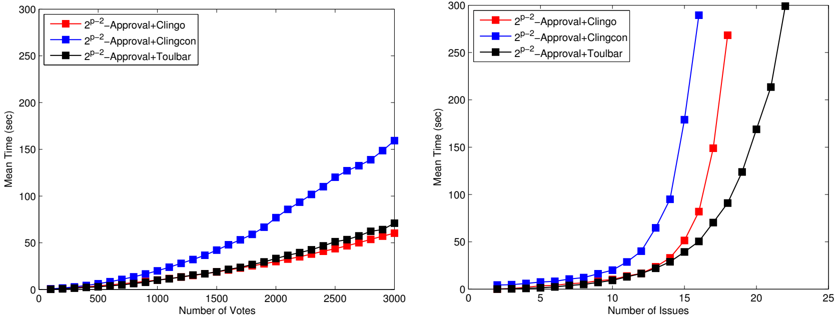
$0_j 0_k$	$0_s > 1_s$
$0_j 1_k$	$1_s > 0_s$
$1_j 0_k$	$0_s > 1_s$
$1_j 1_k$	$1_s > 0_s$

Figure: Random LP tree

Experimentation

- The winner problem:
  - fix the number of issues (10) and increase the number of votes in a profile (up to 3000)
  - fix the number of votes in a profile (1000) and increase the number of issues (up to 25)
- The evaluation problem: for a randomly generated profile (1000 votes), computed winning score WS and solved the evaluation problem with various *thresholds* (percentages of WS): $\{5\%{\cdot}WS, 10\%{\cdot}WS, \ldots, 100\%{\cdot}WS, WS+1\}$

Varying p and n: 2^{p-2}-approval

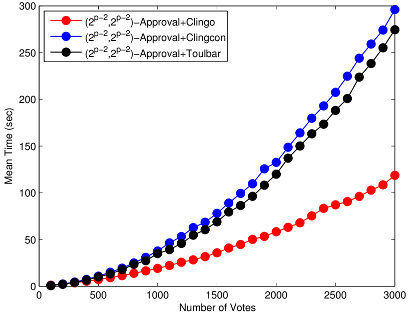


(a) Fixed #issues (10)

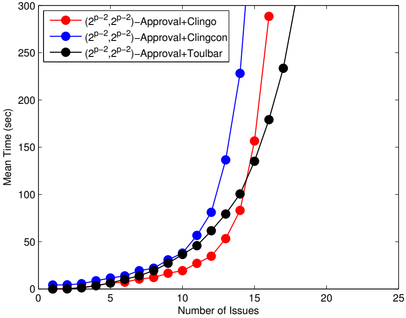
(b) Fixed #votes (1000)

Figure: Solving the winner problem

Varying p and n: $(2^{p-2}, 2^{p-2})$-approval [7]



(a) Fixed #issues (10)

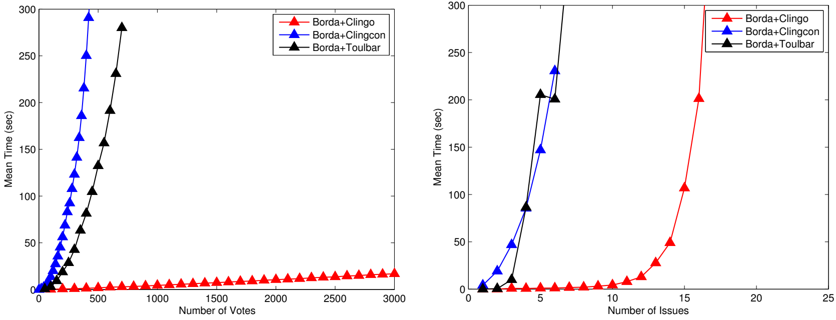


(b) Fixed #votes (1000)

Figure: Solving the winner problem

[7] scoring vector: $(2, \ldots, 2, 1, \ldots, 1, 0, \ldots, 0)$ with the numbers of 2's and 1's equal to 2^{p-2}

Varying p and n: Borda



(a) Fixed #issues (10)

(b) Fixed #votes (1000)

Figure: Solving the winner problem

Conclusion

1. The formalism of *lexicographic preference trees* is a **concise** representation of preferences over combinatorial domains
   - natural way to express preferences
   - induce total orders
   - easily compute rank of a given alternative
2. Computational complexity of the winner and evaluation problems has yet been fully studied
   - polynomial time algorithms
3. ASP and W-MAXSAT are **effective** in modeling and solving the problems even for 1000 votes over up to 22 issues (about 4 million alternatives)
   - W-MAXSAT solver *toulbar* is better than ASP solvers for 2^{p-2}-approval and $(2^{p-2}, 2^{p-2})$-approval
   - ASP solvers *clingo* and *clingcon* are better than *toulbar* for Borda

Future Work

1. Generate richer classes of random LP trees
2. Aggregate preferences in other formalisms, such as conditional preference networks (CP-nets) [8] and answer set optimization preferences [9]

The slides are available at www.cs.uky.edu/∼liu

Thank you!

[8] Boutilier, Brafman, Domshlak, Hoos, and Poole. *CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements*, 2004.

[9] Brewka, Niemela, and Truszczynski. *Answer set optimization*, 2003.

Appendix: LP Trees vs Total Orders

LP Trees and Total Orders

Every LP tree over $\mathcal{I}$ can be represented by a total order over $\mathcal{X}$, but not vice versa.

For example, $00 \succ 11 \succ 01 \succ 10$ cannot be translated to an LP tree.

The Score Problem

Let r be a positional scoring rule with a scoring vector w. Given a profile P of n LP trees over p issues, an alternative $o \in \mathcal{X}$ and a positive integer T, the *score* problem is to decide whether $s_w(P, o) \geq T$.

For all positional scoring rules, we have

	UP	CP
UI	$O(np)$	$O(np)$
CI	$O(np)$	$O(np)$

Figure: The score problem [10]

10 Jérôme Lang, Jérôme Mengin, and Lirong Xia. *Aggregating conditionally lexicographic preferences on multi-issue domains*, 2012.

Appendix: Effective Implicit

Effective Implicit Positional Scoring Rules

Let r be a positional scoring rule, and w its underlying scoring vector. Rule r is *effective implicit* if, given m and i ($1 \leq i \leq m$), there is an algorithm that computes the value w_i in time polynomial in the size of m.

- Borda: $w_i = m - i$
- k-approval: if $i \leq k$, $w_i = 1$; otherwise, $w_i = 0$
- (k, l)-approval: if $i \leq k$, $w_i = a$; if $k \leq i \leq l$, $w_i = b$; otherwise, $w_i = 0$