Aggregating Conditionally Lexicographic Preferences Using Answer Set Programming Solvers

XUDONG LIU
Department of Computer Science, University of Kentucky, USA
liu@cs.uky.edu

HIGHLIGHT

We consider voting over combinatorial domains, where alternatives are binary tuples. We assume that votes are specified as ***conditionally lexicographic preferences***, or *LP trees*. We study aggregation problems of LP tree votes for several positional scoring rules. Our main goal is to demonstrate that ***answer-set programming*** can be effective in solving the *winner* and the *evaluation* problems for instances of practical sizes.

Voting in Combinatorial Domains

Denote by $\mathcal{I}$ the set of *issues* $\mathcal{I} = \{X_1, X_2, \dots, X_p\}$ ($D(X_i) = \{0_i, 1_i\}$). The space of *alternatives* $\mathcal{X}$ is given by $\mathcal{X} = D(X_1) \times D(X_2) \times \dots \times D(X_p)$. A *vote* is a strict total order on $\mathcal{X}$ and a *profile* is a finite collection of votes.
A *positional scoring rule* is determined by a *scoring vector* $w = (w_0, \dots, w_{m-1})$ of natural numbers, where $w_0 \geq w_1 \geq \dots \geq w_{m-1}$ and $w_0 > w_{m-1}$. Given a vote $v = o_0 > o_1 > \dots > o_{m-1}$, the score of alternative o_i in vote v is w_i, denoted by $s_w(v, o_i)$. Let $\mathcal{V}$ be a profile of votes v, the score of alternative o_i in profile $\mathcal{V}$:

$$s_w(\mathcal{V}, o_i) = \textstyle\sum_{v \in \mathcal{V}} s_w(v, o_i).$$

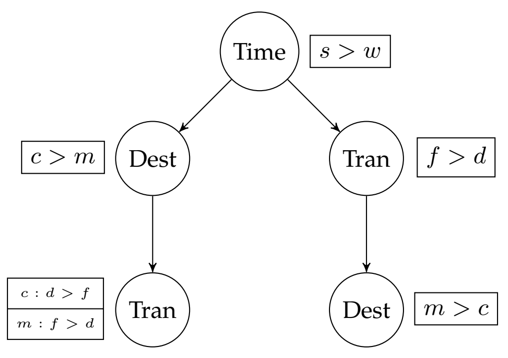
In our work, we focus on three classes of positional scoring rules:

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

Conditionally Lexicographic Preferences

The space of alternatives in combinatorial domains can be huge, even if the set of issues is moderate. It is infeasible for human beings to enumerate all the alternatives from the most prefered one to the least preferred. Conditionally lexicographic preferences (LP-trees) [1] exploit the way people express their preferences over alternatives in combinatorial domains, that is, people may first consider the *most* important issue to them and give preference on that issue, and consider the *next* most important one and so on.
An LP tree over $\mathcal{I}$ consists of three components: (1) a *binary tree*, where each node is labeled by an issue in $\mathcal{I}$ such that each issue appears **exactly once** on each path from the root to a leaf; (2) a *parent* function $\mathcal{P}$ that assigns to each node t a set of "parents" that are nodes whose values determine local preferences associated with node t; and (3) for each node t, a *conditional preference table CPT*(t) that specifies local preferences for the issue labeling t, say X_i, by expressions $u : 1_i > 0_i$ or $u : 0_i > 1_i$, where u is a combination of values of issues labeling $\mathcal{P}(t)$.

Example 1 *A group of friends in Lexington want to make vacation plans for 2014. Having brainstormed for a while, the group decided to focus on three binary issues. The Time of the travel could be either summer (s) or winter (w), the Destination could be either Chicago (c) or Miami (m) and the mode of Transportation could be to drive (d) or fly (f). Jane, a member of the group, describe here preferences over vacation options as the following LP tree. Jane's most preferred vacation option is easy to compute, that is, a summer vacation to Chicago by driving.*



Problems and Computational Complexity

Definition 1 *Let r be a positional scoring rule with a scoring vector w. Given a profile $\mathcal{V}$ of n LP trees over p issues, the **winner problem** is to compute an alternative $o \in \mathcal{X}$ with the maximum score $s_w(\mathcal{V}, o)$.*

Definition 2 *Let r be a positional scoring rule with a scoring vector w. Given a profile $\mathcal{V}$ 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 $s_w(\mathcal{V}, o) \geq R$.*

For Borda, Lang, Mengin, and Xia [2] proved:

	CP	UP
CI	NPC	NPC
UI	NPC	P

(1). Evaluation

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

(2). Winner

The story for k-approval is more complex [2]. If k is a constant independent of n and p or $k = 2^{p-1}$, the winner and the evaluation problems are in P. If $k = c \cdot 2^{p-M}$ (c and M are constants) and $k \neq 2^{p-1}$, we have

	CP	UP
CI	NPC	NPC
UI	NPC	NPC

(3). Evaluation

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

(4). Winner

We obtained new complexity results for special classes of (k, l)-approval. When $k = l = 2^{p-2}$ or $k = l = 2^{p-3}$, we have

	CP	UP
CI	NPC	NPC
UI	NPC	NPC

(5). Evaluation

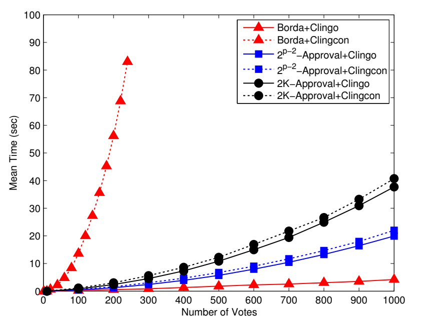
	CP	UP
CI	NP-Hard	NP-Hard
UI	NP-Hard	NP-Hard

(6). Winner

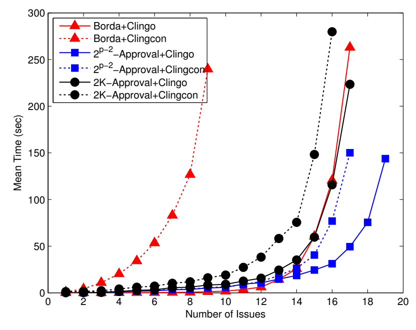
Solving the Problems Using ASP

We provided insight into effectiveness of *Anser-Set Programming* (ASP) [3] in modeling and solving the winner and evaluation problems. ASP is a language for modeling search and optimization problems defined by constraints. We encoded LP trees as logic programs. We also encoded the winner and the evaluation problems in two ASP dialects customized for two ASP solvers, *clingo* and *clingcon*. *Answer sets* computed correspond to solutions to the two aggregation problems. To experiment with profiles of LP trees, we developed methods to generate **random** LP tree of size linear in the number of issues.

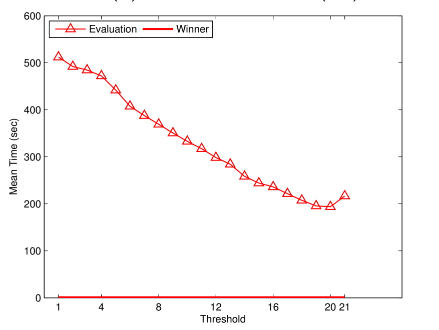
Experimentation Results



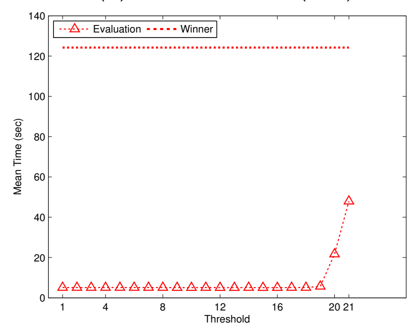
(a). Fixed #issues (10)



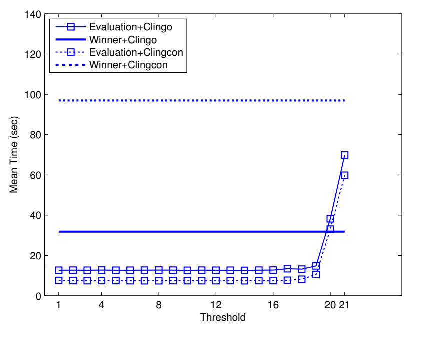
(b). Fixed #votes (500)



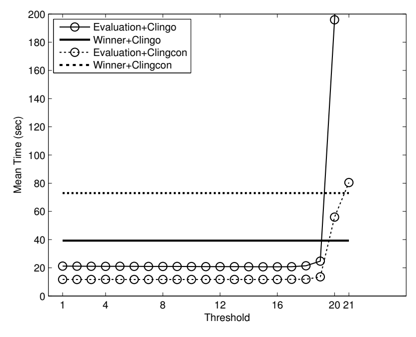
(c). Borda,*clingo* (500v/3i)



(d). Borda,*clingcon* (500v/8i)



(e). 2^{p-2}-approval (500v/16i)



(f). $2K$-approval (500v/14i)

References

[1] Richard Booth, Yann Chevaleyre, Jérôme Lang, Jérôme Mengin, and Chattrakul Sombattheera. Learning conditionally lexicographic preference relations. In *ECAI*, pages 269–274, 2010.

[2] Jérôme Lang, Jérôme Mengin, and Lirong Xia. Aggregating conditionally lexicographic preferences on multi-issue domains. In *CP*, pages 973–987, 2012.

[3] V.W. Marek and M. Truszczynski. Stable models and an alternative logic programming paradigm. In K.R. Apt, V.W. Marek, M. Truszczynski, and D.S. Warren, editors, *The Logic Programming Paradigm: a 25-Year Perspective*, pages 375–398. Springer, Berlin, 1999.