Modeling, Learning and Reasoning about Preference Trees over Combinatorial Domains

Xudong Liu

Advisor: Dr. Miroslaw Truszczynski

Department of Computer Science
College of Engineering
University of Kentucky
Lexington, KY, USA
Tuesday, 4/19/2016

Preferences Are Ubiquitous

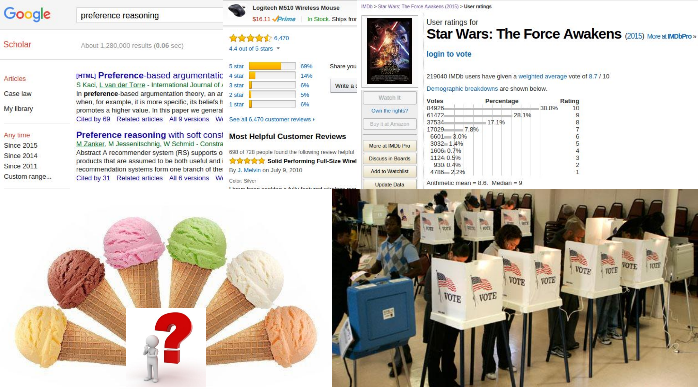


Figure : Preferences of different forms

Describing Preferences

Car1 | Car2

<mvan, 7m, gray, big, honda, med, med> | <sedan, 5, blue, med, vw, med, med>

Figure : How to express preferences?

1. How will I rate cars?
    - For BodyType, I will assign 7 points to minivans, 5 to sedans, ...
    - For Color, I will assign 8 points to blue, 4 to gray, ...
2. What are the desired properties I see in cars?
    - I prefer minivans to sedans, ...
    - If minivan, I prefer gray to blue; if sedan, I prefer blue to gray; ...

Describing Preferences

Car1

<mvan, 7m, gray, big, honda, med, med>

Car2

<sedan, 5, blue, med, vw, med, med>

Figure : How to express preferences?

1. How will I rate cars? (**Quantitative**)
   - For BodyType, I will assign 7 points to minivans, 5 to sedans, ...
   - For Color, I will assign 8 points to blue, 4 to gray, ...
2. What are the desired properties I see in cars? (**Qualitative**)
   - I prefer minivans to sedans, ...
   - If minivan, I prefer gray to blue; if sedan, I prefer blue to gray; ...

Combinatorial Domains

Combinatorial Domains

Let $\mathcal{I}$ be a finite set of attributes $\{X_1, \ldots, X_p\}$, associated with a set of finite domains $\{Dom(X_1), \ldots, Dom(X_p)\}$ for each attribute X_i. A *combinatorial domain* $CD(\mathcal{I})$ is a set of *objects* described by combinations of values from $Dom(X_i)$:

$$CD(\mathcal{I}) = \prod_{X_i \in \mathcal{I}} Dom(X_i).$$

Combinatorial Domains: Example

Domain of cars over set $\mathcal{I}$ of p binary attributes:

1. **BodyType**: {mvan, sedan}.
2. **Capacity**: {5, 7m}.
3. **Color**: {blue, gray}.

$\vdots$

$$CD(\mathcal{I}) = \underbrace{\{\langle \text{sedan, 5, blue, } \ldots \rangle, \langle \text{mvan, 7m, gray, } \ldots \rangle, \ldots\}}_{2^p \text{ objects, too many!}}.$$

Combinatorial Domains: Example

Domain of cars:

1. **BodyType**: {mvan, sedan, sport, suv}.
2. **Capacity**: {2, 5, 7m}.
3. **Color**: {black, blue, gray, red, white}.
4. **LuggageSize**: {big, med, small}.
5. **Make**: {bmw, ford, honda, vw}.
6. **Price**: {low, med, high, vhigh}.
7. **Safety**: {low, med, high}.

Single Agent

Car1 Car2

<mvan, 7m, gray, big, honda, med, med> <sedan, 5, blue, med, vw, med, med>

Figure : Dominance and Optimization

Multi-Agent

Figure : Social Choice and Welfare

Research Problems of Interest

1. Preference representation formalisms to compactly model qualitative preferences over combinatorial domains.
2. Preference elicitation and learning methods to cast preferences of agents as a theory in a preference formalism.
3. Preference reasoning tasks:
   - Dominance and optimization
   - Preference aggregation

Preference Modeling

Q: How do we compactly represent qualitative preferences over combinatorial domains?

1. Preference Trees (P-trees)[1,11]
2. Partial Lexicographic Preference Trees (PLP-trees)[8]
3. Lexicographic Preference Trees (LP-trees)[4,13]

---

[1] Niall M Fraser. "Ordinal preference representations". In: Theory and Decision (1994)

[2] Xudong Liu and Miroslaw Truszczynski. "Preference Trees: A Language for Representing and Reasoning about Qualitative Preferences". In: Proceedings of the 8th Multidisciplinary Workshop on Advances in Preference Handling (MPREF). 2014

[3] Xudong Liu and Miroslaw Truszczynski. "Learning Partial Lexicographic Preference Trees over Combinatorial Domains". In: Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI). 2015

[4] Richard Booth et al. "Learning conditionally lexicographic preference relations". In: ECAI. 2010

[5] Xudong Liu and Miroslaw Truszczynski. "Aggregating Conditionally Lexicographic Preferences Using Answer Set Programming Solvers". In: Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT). 2013

Preference Learning

Q: How do we learn predictive qualitative preference models over combinatorial domains?

1. Partial Lexicographic Preference Trees (PLP-trees)[6,7,8]
   - Active and passive learning
   - Compute a (possibly small) PLP-tree consistent with all the data
   - Compute a PLP-tree that agrees with the data as much as possible
2. Empirical Learning of PLP-trees and PLP-forests[9]

[6] Michael Schmitt and Laura Martignon. "On the complexity of learning lexicographic strategies". In: The Journal of Machine Learning Research (2006)

[7] József Dombi, Csanád Imreh, and Nándor Vincze. "Learning lexicographic orders". In: European Journal of Operational Research (2007)

[8] Xudong Liu and Miroslaw Truszczynski. "Learning Partial Lexicographic Preference Trees over Combinatorial Domains". In: Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI). 2015

[9] Xudong Liu and Miroslaw Truszczynski. "Learning Partial Lexicographic Preference Trees and Forests over Multi-Valued Attributes". In: Review by ECAI-16 Program Committee

Preference Reasoning

Q: How do we reason about preferences over combinatorial domains?

1. Preference Optimization[10,11,12]:
   - Dominance testing: $o_1 \succ_P o_2$?
   - Optimality testing: $o_1 \succ_P o_2$ for all $o_2 \neq o_1$?
   - Optimality computing: what is the optimal outcome wrt P?
2. Preference Aggregation[13]:
   - Winner determination: which candidate wins the election?
   - "Strong" candidate: a candidate with score more than a threshold?

---

[10] Jérôme Lang, Jérôme Mengin, and Lirong Xia. "Aggregating Conditionally Lexicographic Preferences on Multi-issue Domains". In: CP. 2012

[11] Xudong Liu and Miroslaw Truszczynski. "Preference Trees: A Language for Representing and Reasoning about Qualitative Preferences". In: Proceedings of the 8th Multidisciplinary Workshop on Advances in Preference Handling (MPREF). 2014

[12] Xudong Liu and Miroslaw Truszczynski. "Reasoning with Preference Trees over Combinatorial Domains". In: Proceedings of the 4th International Conference on Algorithmic Decision Theory (ADT). 2015

[13] Xudong Liu and Miroslaw Truszczynski. "Aggregating Conditionally Lexicographic Preferences Using Answer Set Programming Solvers". In: Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT). 2013

Outline

1. Modeling qualitative preferences:
   - Preference trees (P-trees)
   - Partial lexicographic preference trees (PLP-trees)
2. Learning PLP-trees and PLP-forests
3. Aggregating LP-trees
4. Future research directions

Outline

1. Modeling qualitative preferences:
   - Preference trees (P-trees)
   - Partial lexicographic preference trees (PLP-trees)
2. Learning PLP-trees and PLP-forests
3. Aggregating LP-trees
4. Future research directions

Preference Trees

1. Let $\mathcal{I} = \{X_1, \ldots, X_p\}$ be a set of attributes, and $D(\mathcal{I}) = \{Dom(X_1), \ldots, Dom(X_p)\}$ a set of finite domains for $\mathcal{I}$.
2. A *literal* is an assignment to an attribute. We denote by $X_i := x_{i,j}$ the literal that assigns value $x_{i,j} \in Dom(X_i)$ to X_i. When no confusion, we write $x_{i,j}$, instead of $X_i := x_{i,j}$, as a literal. We then denote by $\mathcal{L} = \{x_{i,j} \in Dom(X_i) : X_i \in \mathcal{I}\}$ the set of literals given $\mathcal{I}$ and $D(\mathcal{I})$.
3. The combinatorial domain $CD(\mathcal{I})$ is defined as earlier.

Preference Trees

4. A **P-tree** T over $CD(\mathcal{I})$ is a binary tree, where non-leaf nodes are labeled with propositional formulas over $\mathcal{L}$.
5. Given an outcome $o \in CD(\mathcal{I})$, the **leaf** $l_T(o)$ is the leaf reached by traversing the tree T according to o. When at a node N labeled with φ, if $o \models \varphi$, we descend to the left child of N; otherwise, to the right.
6. For $o_1, o_2 \in CD(\mathcal{I})$, we have $o_1 \succ_T o_2$ if $l_T(o_1) \succ_T l_T(o_2)$, and $o_1 \approx_T o_2$ if $l_T(o_1) = l_T(o_2)$. Outcome o_1 is **optimal** if there exists no o_2 such that $o_2 \succ_T o_1$.

Preference Trees (P-Trees)

Let φ, ψ, and π be propositional formulas over the set $\mathcal{L}$ of literals that are values from $\bigcup_{X_i \in V} Dom(X_i)$.

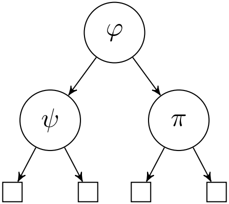


Figure : A P-tree

$$\varphi \wedge \psi \succ \varphi \wedge \neg\psi \succ \neg\varphi \wedge \pi \succ \neg\varphi \wedge \neg\pi.$$

Preference Trees (P-Trees)

Let φ, ψ, and π be propositional formulas over the set $\mathcal{L}$ of literals that are values from $\bigcup_{X_i \in V} Dom(X_i)$.

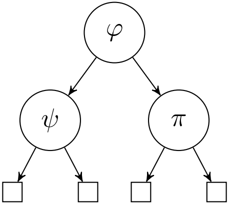


Figure : A P-tree

$$\varphi \wedge \psi \succ \varphi \wedge \neg\psi \succ \neg\varphi \wedge \pi \succ \neg\varphi \wedge \neg\pi.$$

Total preorder

Example: The Cars Domain

1. **BodyType**(X_1): {mvan($x_{1,1}$), sedan($x_{1,2}$), sport($x_{1,3}$), suv($x_{1,4}$)}.
2. **Capacity**(X_2): {2, 5, 7m}.
3. **Color**(X_3): {black, blue, gray, red, white}.
4. **LuggageSize**(X_4): {big, med, small}.
5. **Make**(X_5): {bmw, ford, honda, vw}.
6. **Price**(X_6): {low, med, high, vhigh}.
7. **Safety**(X_7): {low, med, high}.

Example: Preference Trees over Cars

BodyType(X_1): {mvan($x_{1,1}$), sedan($x_{1,2}$), sport($x_{1,3}$), suv($x_{1,4}$)}.
Color(X_3): {black, blue, gray, red, white}.
Price(X_6): {low, med, high, vhigh}.

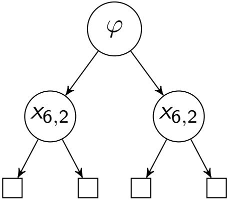


Figure : A P-tree over cars[14]

[14] $\varphi = (x_{1,1} \wedge x_{3,5}) \vee (x_{1,2} \wedge x_{3,2})$.

Example: Preference Trees over Cars

BodyType(X_1): {mvan($x_{1,1}$), sedan($x_{1,2}$), sport($x_{1,3}$), suv($x_{1,4}$)}.
Color(X_3): {black, blue, gray, red, white}.
Price(X_6): {low, med, high, vhigh}.

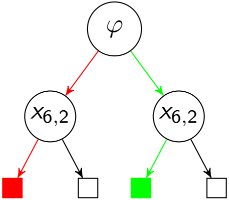


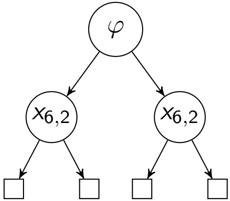
Figure : A P-tree over cars[14]

$Car2 \succ Car1$

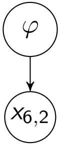
[14] $\varphi = (x_{1,1} \wedge x_{3,5}) \vee (x_{1,2} \wedge x_{3,2})$.

Compact Representation of P-trees

BodyType(X_1): {mvan($x_{1,1}$), sedan($x_{1,2}$), sport($x_{1,3}$), suv($x_{1,4}$)}.
Color(X_3): {black, blue, gray, red, white}.
Price(X_6): {low, med, high, vhigh}.



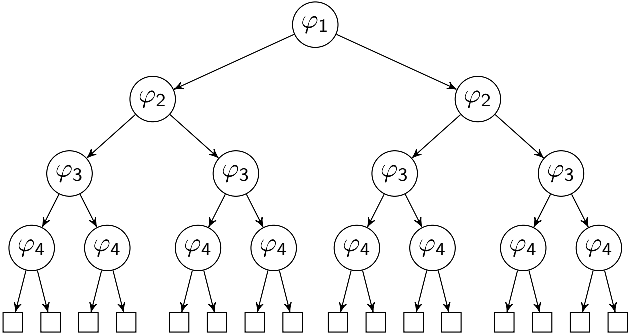
(a) Full



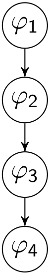
(b) Compact

Figure : Compact P-trees

Compact Representation of P-trees



(a) Full



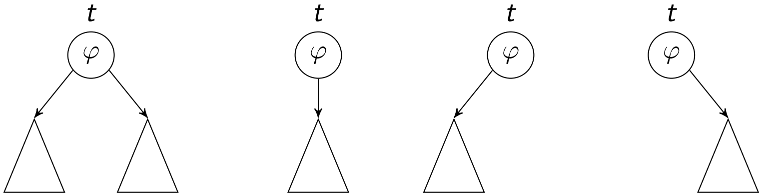
(b) Compact

Figure : Compact P-trees

Compact Representation of P-trees

A *compact P-tree* over $CD(\mathcal{I})$ is a binary tree where

1. every node is labeled with a Boolean formula over $\mathcal{I}$, and
2. every non-leaf node t labeled with φ has either two outgoing edges (Fig. (a)), or one outgoing edge pointing straight-down (Fig. (b)), left (Fig. (c)), or right (Fig. (d)).



(a) (b) (c) (d)

Figure : Compact P-trees

Relative Expressivity of Preference Languages

$$
\begin{array}{ccccc}
 & & \textbf{LP-trees} & & \\
 & & \cap & & \\
 & & \textbf{PLP-trees} & & \\
 & & \cap & & \\
\text{Poss-theories} = \text{ASO-rules} & \subset & \textbf{P-trees} & \subset & \text{ASO-theories}
\end{array}
$$

Computational Complexity Results

Dominance-testing ($\textsc{DomTest}$): $o_1 \succ_T o_2$?
Optimality-testing ($\textsc{OptTest}$): o optimal w.r.t T?
Optimality-with-property ($\textsc{OptProp}$): is there optimal o with property α?

1. $\textsc{DomTest} \in P$
2. $\textsc{OptTest} \in coNP$-complete:
   - The complement problem is reduced from the SAT problem.
3. $\textsc{OptProp} \in \Delta_2^P$-complete:
   - The problem is reduced from the Maximum Satisfying Assignment (MSA) problem.

Outline

1. Modeling qualitative preferences:
   - Preference trees (P-trees)
   - Partial lexicographic preference trees (PLP-trees)
2. Learning PLP-trees and PLP-forests
3. Aggregating LP-trees
4. Future research directions

The Cars Domain

1. **BodyType**(B): {mvan, sedan, sport, suv}.
2. **Capacity**(C): {2, 5, 7m}.
3. **Color**(O): {black, blue, gray, red, white}.
4. **LuggageSize**(L): {big, med, small}.
5. **Make**(M): {bmw, ford, honda, vw}.
6. **Price**(P): {low, med, high, vhigh}.
7. **Safety**(S): {low, med, high}.

Partial Lexicographic Preference Trees (PLP-Trees)

A *PLP-tree* over $CD(\mathcal{I})$ is a tree, where

1. every non-leaf node t is labeled with an attribute $Attr(t)$ in $\mathcal{I}$,
2. every non-leaf node t has $|Dom(Attr(t))|$ outgoing edges labeled with a value of $Attr(t)$, and
3. every attribute appears *at most* once on every branch.

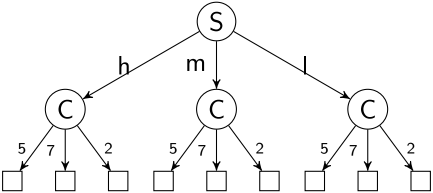


Figure : A PLP-tree over cars

Partial Lexicographic Preference Trees (PLP-Trees)

A *PLP-tree* over $CD(\mathcal{I})$ is a tree, where

1. every non-leaf node t is labeled with an attribute $Attr(t)$ in $\mathcal{I}$,
2. every non-leaf node t has $|Dom(Attr(t))|$ outgoing edges labeled with a value of $Attr(t)$, and
3. every attribute appears *at most* once on every branch.

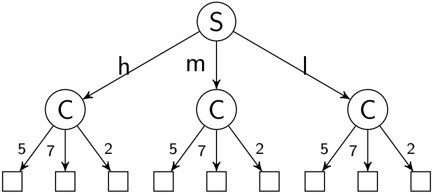


Figure : A PLP-tree over cars

Total preorder

Partial Lexicographic Preference Trees (PLP-Trees)

Car1

<mvan, 7m, gray, big, honda, med, med>

Car2

<sedan, 5, blue, med, vw, med, med>

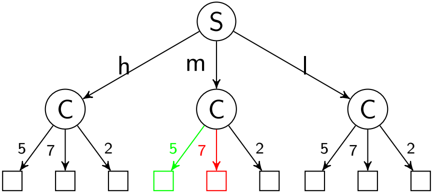
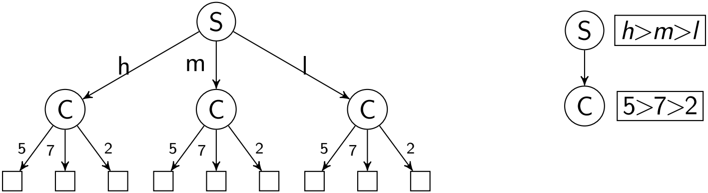


Figure : A PLP-tree over cars

$Car2 \succ Car1$

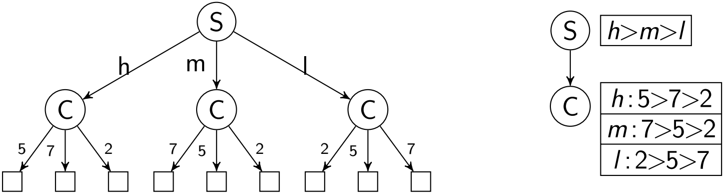
Compact Representations of PLP-Tree



(a) Full (b) Compact

Figure : Unconditional Importance & Unconditional Preference (UIUP)

Compact Representations of PLP-Tree



(a) Full (b) Compact

Figure : Unconditional Importance & Conditional Preference (UICP)

Compact Representations of PLP-Tree

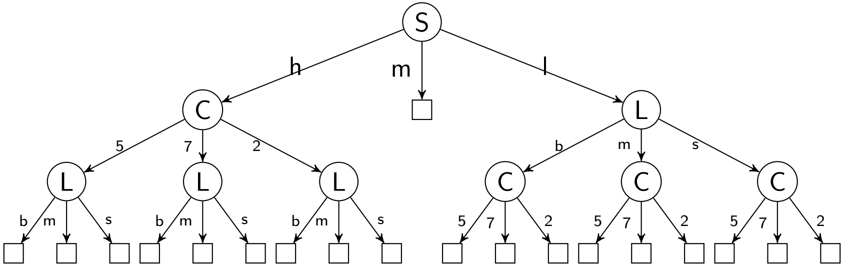


Figure : Conditional Importance & Unconditional Preference (CIUP)

Compact Representations of PLP-Tree

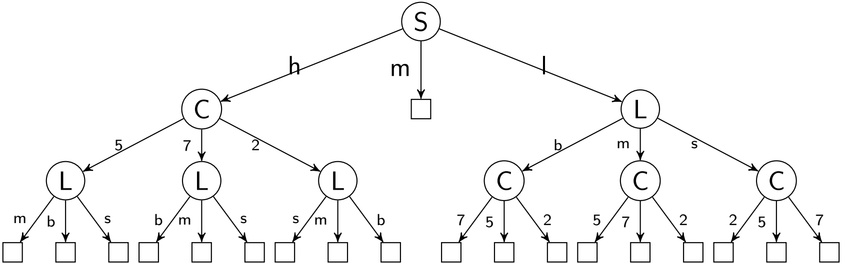


Figure : Conditional Importance & Conditional Preference (CICP)

Lexicographic Preference Trees (LP-Trees)

1. An *LP-tree* $\mathcal{L}$ over $CD(\mathcal{I})$ is a PLP-tree, where
   - each attribute appears **exactly once** on every path from the root to a leaf.
   - Unlike PLP-trees, an LP-tree induces a total order.

Conclusion

1. Generalizing LP-trees, PLP-trees compactly represent total *preorders* over combinatorial domains, by allowing agents to specify, on each path, only a subset of attributes (i.e., those useful ones).
2. P-trees further generalize PLP-trees by labeling the nodes with *propositional formulas*, in practice, usually built with small number (e.g., at most 3) of attributes.

Outline

1. Modeling qualitative preferences:
   - Preference trees (P-trees)
   - Partial lexicographic preference trees (PLP-trees)
2. Learning PLP-trees and PLP-forests
3. Aggregating LP-trees
4. Future research directions

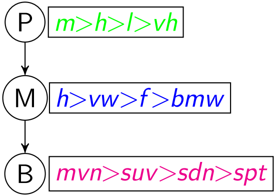
Consistent Learning (ConsLearn)

Given an example set $\mathcal{E}$, decide whether there exists a PLP-tree T (of a particular type) such that T is consistent with $\mathcal{E}$.

(<sdn,5,blk,m,h,m,m>,<suv,7m,wht,b,f,m,m>)

(<spt,2,wht,s,bmw,h,h>,<spt,2,wht,s,bmw,vh,h>)

(<mvn,7m,gry,b,f,m,m>,<sdn,5,bl,m,f,m,m>)



UIUP tree

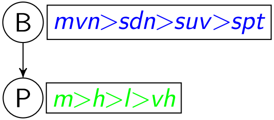
Small Learning (SmallLearn)

Given an example set $\mathcal{E}$ and a positive integer l ($l \leq |\mathcal{E}|$), decide whether there exists a PLP-tree T (of a particular type) such that T is consistent with $\mathcal{E}$ and $|T| \leq l$.

(<sdn,5,blk,m,h,m,m>,<suv,7m,wht,b,f,m,m>)

(<spt,2,wht,s,bmw,h,h>,<spt,2,wht,s,bmw,vh,h>)

(<mvn,7m,gry,b,f,m,m>,<sdn,5,bl,m,f,m,m>)



UIUP tree

Maixmal Learning (MaxLearn)

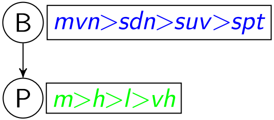
Given an example set $\mathcal{E}$ and a positive integer k ($k \leq m$), decide whether there exists a PLP-tree T (of a particular type) such that T satisfies at least k examples in $\mathcal{E}$.

(<sdn,5,blk,m,h,m,m>,<suv,7m,wht,b,f,m,m>)

(<spt,2,wht,s,bmw,h,h>,<spt,2,wht,s,bmw,vh,h>)

(<mvn,7m,gry,b,f,m,m>,<sdn,5,bl,m,f,m,m>)

(<suv,7m,gry,b,vw,vh,m>,<suv,7m,gry,b,vw,h,m>)



UIUP tree

Complexity Results on PLP-trees

	UP	CP
UI	P	NP
CI	NPC[15]	P

(a) ConsLearn

	UP	CP
UI	NPC	NPC
CI	NPC	NPC

(b) SmallLearn

	UP	CP
UI	NPC[16]	NPC
CI	NPC	NPC

(c) MaxLearn

Figure : Complexity results for learning PLP-trees

[15] Booth et al., *Learning Conditionally Lexicographic Preference Relations*, 2010.

[16] Schmitt and Martignon, *On the Complexity of Learning Lexicographic Strategies*, 2006.

Experimentation

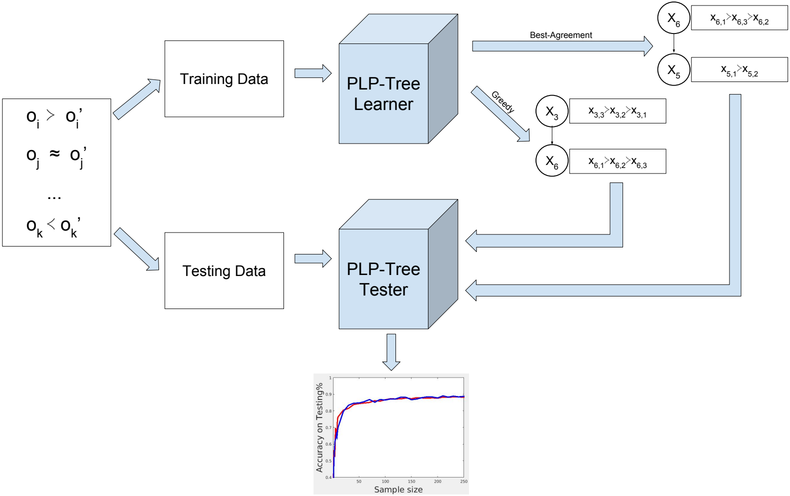


Figure : PLP-tree learning system

Datasets

Dataset	p	$\lvert\mathcal{X}\rvert$	$\lvert\mathcal{E}^{\succ}\rvert$	$\lvert\mathcal{E}^{\approx}\rvert$
BreastCancerWisconsin	9	270	9,009	27,306
CarEvaluation	6	1,728	682,721	809,407
CreditApproval	10	520	66,079	68,861
GermanCredit	10	914	172,368	244,873
Ionosphere	10	118	3,472	3,431
MammographicMass	5	62	792	1,099
Mushroom	10	184	8,448	8,388
Nursery	8	1,266	548,064	252,681
SPECTHeart	10	115	3,196	3,359
TicTacToe	9	958	207,832	250,571
Vehicle	10	455	76,713	26,572
Wine	10	177	10,322	5,254

Figure : Preference Learning Library[17]

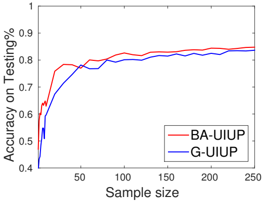
[17] `http://www.cs.uky.edu/~liu/preflearnlib.php`

Experimental Results: Best-Agreement vs. Greedy

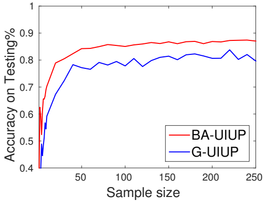
Dataset	BA-UIUP	G-UIUP
BCW	88.4	88.2
CE	84.8	83.6
CA	91.1	89.3
GC	72.2	72.2
IN	87.0	79.6
MM	87.5	86.8
MS	84.8	70.3
NS	91.8	91.7
SH	93.2	92.6
TTT	72.1	71.9
VH	76.8	76.6
WN	96.0	95.5

Table : Accuracy (percentage of correctly handled testing examples) for UIUP PLP-trees learned using the best-agreement and the greedy methods on the learning data (250 of $\mathcal{E}^{\succ}$)

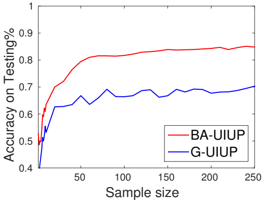
Experimental Results: Best-Agreement vs. Greedy



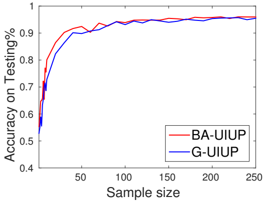
(a) CarEvaluation



(b) Ionosphere



(c) Mushroom



(d) Wine

Figure : Learning curves solving MaxLearn for UIUP PLP-trees

Experimental Results: Greedy

Dataset	UIUP	UICP-1	CIUPB	CIUPD	CICP
BCW	90.7	91.4	91.0	90.7	91.4
CE	85.8	86.0	85.8	85.9	86.0
CA	91.4	91.7	91.6	92.0	92.2
GC	74.3	74.6	74.3	74.5	75.7
IN	87.1	86.9	87.2	88.5	90.4
MM	88.2	89.5	87.3	86.9	90.0
MS	71.6	74.2	77.1	75.6	76.6
NS	92.9	93.0	93.0	93.0	93.0
SH	93.4	94.9	95.4	94.8	95.7
TTT	73.9	74.5	74.4	75.4	76.2
VH	79.2	80.4	80.3	80.0	81.2
WN	95.5	97.8	97.8	97.5	97.8

Table : Accuracy percents on the testing data (30% of $\mathcal{E}^{\succ}$) for all four classes of PLP-trees, using models learned by the greedy algorithm from the learning data (the other 70% of $\mathcal{E}^{\succ}$)

Experimental Results: Sizes of PLP-trees by Greedy

Dataset	UIUP	UICP-1	CI	$\lvert\mathcal{E}^{\succ}_{train}\rvert$
BCW	9	33	87,381	6,306
CE	6	21	853	477,904
CA	10	37	91,477	46,255
GC	10	37	349,525	120,657
IN	10	19	1,023	2,430
MM	5	17	341	554
MS	10	37	91,477	5,913
NS	8	29	7,765	383,644
SH	10	19	1,023	2,237
TTT	9	25	9,841	145,482
VH	10	37	349,525	53,699
WN	10	37	349,525	7,225

Table : Maximum sizes of trees for all the classes and the training sample sizes for all datasets

Experimental Results: Sizes of PLP-trees by Greedy

Dataset	UIUP	UICP-1	CIUPB	CIUPD	CICP
BCW	6.7	21.8	19.8	28.0	25.7
CE	6.0	17.0	73.2	108.9	109.5
CA	9.0	24.7	31.3	78.6	81.1
GC	9.7	36.0	49.8	210.3	190.0
IN	9.6	17.2	19.8	31.5	30.6
MM	4.5	14.7	8.3	10.8	10.0
MS	7.6	20.7	15.7	22.7	16.3
NS	8.0	25.7	56.2	121.0	116.9
SH	8.4	13.7	13.0	18.4	19.0
TTT	8.0	21.8	36.8	126.8	115.2
VH	9.0	32.7	33.9	101.3	105.4
WN	5.1	13.3	14.2	16.9	14.6

Table : Sizes of trees learned by the greedy algorithm from the training data (70% of $\mathcal{E}^{\succ}$)

Partial Lexicographic Preference Forests (PLP-Forests)

1. Inspired by *random forests*, we proposed *PLP-forests* that are sets of PLP-trees; thus, the four classes.
2. To reduce the overfitting of a PLP-tree, a PLP-forest
   - consists of *diverse* trees (learned from *small* training samples), and
   - aggregates its constituent trees using the *Pairwise Majority Rule* (PMR).

Experimental Results: Best-Agreement vs. Greedy

Dataset	G+Tree	G+Forest	BA+Forest
BCW	90.7	93.4	95.1
CE	85.8	91.9	89.2
CA	91.4	91.5	93.1
GC	74.3	75.4	77.9
IN	87.1	83.0	92.5
MM	88.2	89.1	90.8
MS	71.6	78.8	90.2
NS	92.9	93.2	94.0
SH	93.4	93.7	94.9
TTT	73.9	75.1	77.2
VH	79.2	82.7	81.9
WN	95.5	95.8	96.9

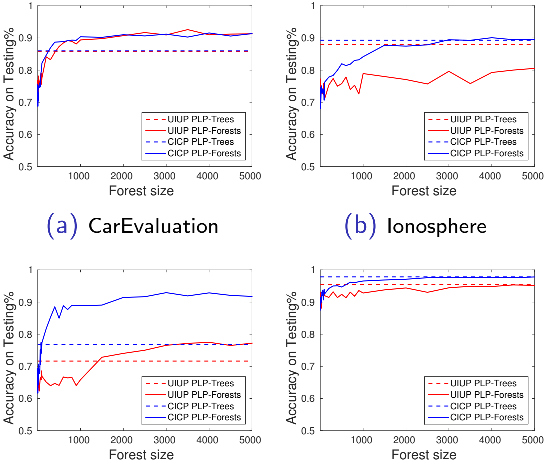
Table : Accuracy percents on the testing data (30% of $\mathcal{E}^{\succ}$) for UIUP trees and forests of 5000 UIUP trees, using the greedy and the best-agreement algorithms from the learning data (the other 70% of $\mathcal{E}^{\succ}$)

Experimental Results: Greedy

Dataset	UIUP	UICP-1	CIUPB	CIUPD	CICP
BCW	93.4	94.1	93.7	94.1	94.0
CE	91.9	88.3	91.4	89.7	91.4
CA	91.5	91.6	92.8	92.9	93.0
GC	75.4	73.8	76.1	76.1	76.2
IN	83.0	87.9	89.3	89.4	89.5
MM	89.1	90.1	90.0	90.1	90.2
MS	78.8	87.2	92.2	92.2	91.8
NS	93.2	89.9	93.3	93.4	93.4
SH	93.7	93.5	93.6	93.6	93.7
TTT	75.1	75.2	76.6	76.5	76.9
VH	82.7	81.8	83.2	83.2	83.4
WN	95.8	95.4	97.5	97.8	97.8

Table : Accuracy percents on the testing data (30% of $\mathcal{E}^{\succ}$) for all four classes of PLP-forests of 5000 trees, using the greedy algorithm from the learning data (the other 70% of $\mathcal{E}^{\succ}$)

Experimental Results: UIUP vs. CICP



(a) CarEvaluation (b) Ionosphere (c) Mushroom (d) Wine

Figure : Greedy learning curves solving MaxLearn for PLP-forests

Conclusion

1. PLP-trees and PLP-forests are *expressive* preference models.
2. PLP-forests aggregated by PRM provide in general *higher* accuracy than PLP-trees.
3. PLP-trees and PLP-forests learned by a greedy approximation method have accuracy *comparable* to best-agreement PLP-trees and PLP-forests.
4. The greedy algorithms are *fast*, can work with *large* datasets (of $\sim$ half million examples), and can compute *small* models.

Outline

1. Modeling qualitative preferences:
   - Preference trees (P-trees)
   - Partial lexicographic preference trees (PLP-trees)
2. Learning PLP-trees and PLP-forests
3. Aggregating LP-trees
4. Future research directions

Positional Scoring Rules

- k-approval: $(1, \ldots, 1, 0, \ldots, 0)$ with k being the number of 1's.
- (k, l)-approval: $(c, \ldots, c, d, \ldots, d, 0 \ldots, 0)$, where c and d are constants ($c > d$), and the numbers of c's and d's equal to k and l.
- b-Borda: $(b, b-1, \ldots, b-m+1)$, where b is a constant and m is the number of candidates.

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 attributes 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 attributes, the *winner* problem is to compute an alternative $o \in \mathcal{X}$ with the maximum score $s_w(o, P)$.

Complexity of the Evaluation Problem: k-Approval

	UP	CP
UI	P	P
CI	P	P

(a) $k = 2^{p-1} \pm f(p)$, $f(p)$ is a poly

	UP	CP
UI	NPC	NPC
CI	NPC	NPC

(b) $k = 2^{p-c}$, $c > 1$ is a const

Figure : k-Approval

Complexity of the Evaluation Problem: (k, l)-Approval

	UP	CP
UI	P	P
CI	P	P

(a) $k = l = 2^{p-1}$

	UP	CP
UI	NPC	NPC
CI	NPC	NPC

(b) $k = l = 2^{p-c}$, $c > 1$ is a const

Figure : (k, l)-Approval

Complexity of the Evaluation Problem: b-Borda

	UP	CP
UI	P	NPC
CI	NPC	NPC

(a) $b = 2^p - 1$

	UP	CP
UI	NPC	NPC
CI	NPC	NPC

(b) $b = 2^{p-c} - 1$, $c \geq 1$ is a const

Figure : b-Borda

Modeling the Problems in ASP

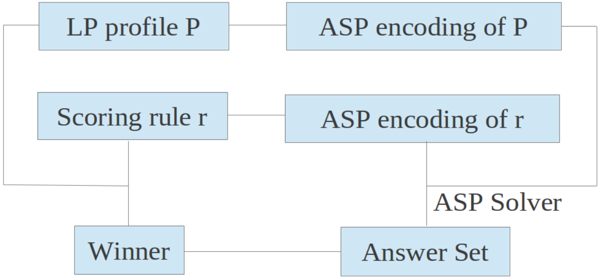


Figure : The winner problem

- Solvers: *clingo*[18], *clingcon*[19]

[18] M. Gebser et al. "Potassco: The Potsdam Answer Set Solving Collection". In: AI Communications (2011)

[19] Max Ostrowski and Torsten Schaub. "ASP modulo CSP: The clingcon system". In: TPLP (2012)

Modeling the Problems in W-MAXSAT

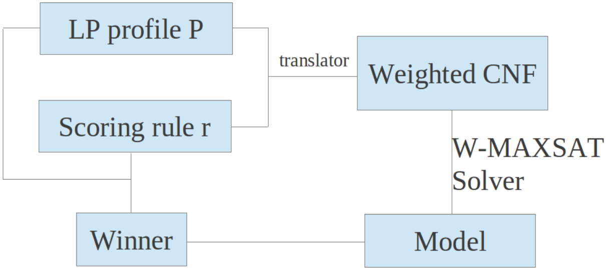
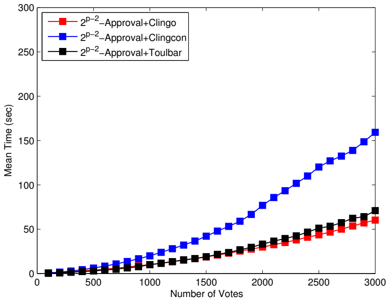


Figure : The winner problem

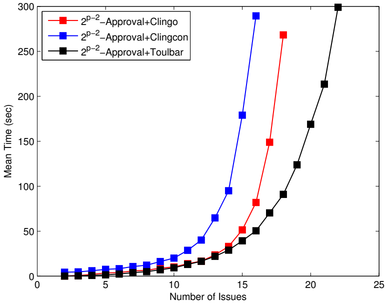
- Solver: *toulbar*[20]

[20] M Sanchez et al. "Max-CSP competition 2008: toulbar2 solver description". In: the Third International CSP Solver Competition (2008)

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



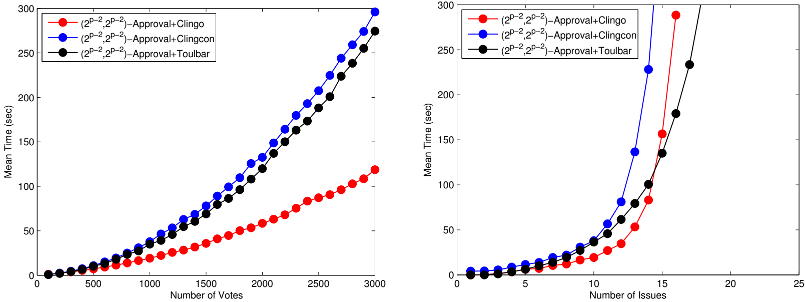
(a) Fixed #attributes (10)



(b) Fixed #votes (1000)

Figure : Solving the winner problem

Varying p and n: $(2^{p-2}, 2^{p-2})$-approval [21]

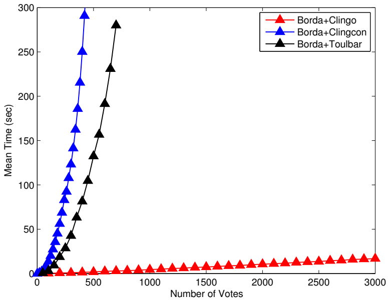


(a) Fixed #attributes (10) (b) Fixed #votes (1000)

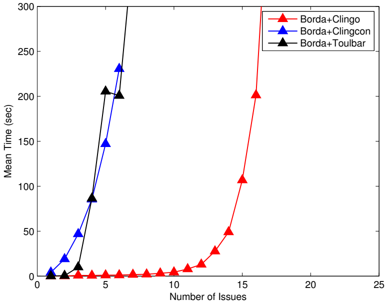
Figure : Solving the winner problem

[21] 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 #attributes (10)



(b) Fixed #votes (1000)

Figure : Solving the winner problem

Conclusion

1. When votes are total orders of candidates, computing a winner for a positional voting rule is computationally easy; not necessarily so, when they are LP-trees over combinatorial domains.
2. For the cases when determining a winner is computationally hard, we solved this problem using ASP and empirically showed that ASP tools are *effective* on large instances.

Outline

1. Modeling qualitative preferences:
   - Preference trees (P-trees)
   - Partial lexicographic preference trees (PLP-trees)
2. Learning PLP-trees and PLP-forests
3. Aggregating LP-trees
4. Future research directions

Data-Driven Preference Engineering

Data-Driven Preference Learning:

1. Recommender Systems[22]:
   - Collaborative
   - Content-based
   - Hybrid
2. Machine Learning (fitting function):
   - Supervised learning (e.g., decision trees, random forests)
   - Label ranking[23]
3. Model-based Learning (learning interpretable decision models):
   - Preference Elicitation (Human-in-the-Loop)
   - Conditional Preference Networks, Preference Trees
   - Stochastic Models (e.g., Choquet integral[24], TOPSIS-like models[25])

---

[22] Gediminas Adomavicius and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions". In: Knowledge and Data Engineering, IEEE Transactions on (2005)

[23] Eyke Hüllermeier et al. "Label ranking by learning pairwise preferences". In: Artificial Intelligence (2008)

[24] Ali Fallah Tehrani, Weiwei Cheng, and Eyke Hüllermeier. "Choquistic Regression: Generalizing Logistic Regression using the Choquet Integral." In: EUSFLAT. 2011

[25] Manish Agarwal, Ali Fallah Tehrani, and Eyke Hüllermeier. "Preference-based Learning of Ideal Solutions in TOPSIS-like Decision Models". In: Journal of Multi-Criteria Decision Analysis (2014)

Data-Driven Preference Engineering

Preference Reasoning and Applications:

1. Social Choice and Welfare[26,27]:
   - Voting
   - Fair division
   - Strategyproof Social Choice
2. Automated Planning and Scheduling[28,29,30]:
   - Travel scheduling
   - Manufacturing
   - Traffic control

---

[26] Kenneth J Arrow, Amartya Sen, and Kotaro Suzumura. Handbook of Social Choice and Welfare. Vol. 1 & 2. 2010

[27] Felix Brandt, Vincent Conitzer, and Ulle Endriss. "Computational social choice". In: Multiagent systems (2012)

[28] Tran Cao Son and Enrico Pontelli. "Planning with preferences using logic programming". In: Theory and Practice of Logic Programming (2006)

[29] Meghyn Bienvenu, Christian Fritz, and Sheila A McIlraith. "Specifying and computing preferred plans". In: Artificial Intelligence (2011)

[30] Hannah Bast et al. "Route planning in transportation networks". In: arXiv preprint (2015)

Related Work

1. Quantitative:
   - Utility/Cost Functions[31]
   - Possibilistic Logic[32]
   - Fuzzy Preference Relations[33]
   - Penalty Logic[34]
2. Qualitative:
   - Answer-Set Optimization Theories[35]
   - Ceteris Paribus Networks (e.g., CP-nets[36], TCP-nets[37], CI-nets[38])
   - Conditional Preference Theories[39]

---

[31] Souhila Kaci. Working with Preferences: Less Is More: Less Is More. Springer Science & Business Media, 2011

[32] Didier Dubois, Jérôme Lang, and Henri Prade. "A Brief Overview of Possibilistic Logic". In: ECSQARU. 1991

[33] SA Orlovsky. "Decision-making with a fuzzy preference relation". In: Fuzzy sets and systems (1978)

[34] Gadi Pinkas. Propositional non-monotonic reasoning and inconsistency in symmetric neural networks. 1991

[35] Gerhard Brewka, Ilkka Niemelä, and Miroslaw Truszczynski. "Answer Set Optimization". In: IJCAI. 2003

[36] C. Boutilier et al. "CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements". In: Journal of Artificial Intelligence Research (2004)

[37] Ronen I. Brafman and Carmel Domshlak. "Introducing Variable Importance Tradeoffs into CP-Nets". In: UAI. 2002

[38] Sylvain Bouveret, Ulle Endriss, and Jérôme Lang. "Conditional importance networks: A graphical language for representing ordinal, monotonic preferences over sets of goods". In: (2009)

[39] Nic Wilson. "Extending CP-Nets with Stronger Conditional Preference Statements". In: AAAI-04. 2004

Summary

1. The languages of P-trees and PLP-trees:
   - P-trees are expressive with labels being propositional formulas, and have an edge over other languages with lower computational complexities for certain decision problems.
   - PLP-trees are P-trees with labels being attributes.
2. Learning PLP-trees and PLP-forests:
   - PLP-trees are highly accurate in modeling preferences arising in practice, and can be effectively learned.
   - PLP-forests, collections of PLP-trees, are empirically shown with reduced overfitting and higher accuracy.
3. Aggregating LP-trees:
   - Preference aggregation problems for LP-trees using positional scoring rules are in general NP-hard.
   - Answer-set programming tools are effective for large instances.

Questions?

Thank you!