ECP

Adaptive Utility Elicitation with Minimax Regret

Paolo Viappiani

Department of Computer Science

University of Toronto

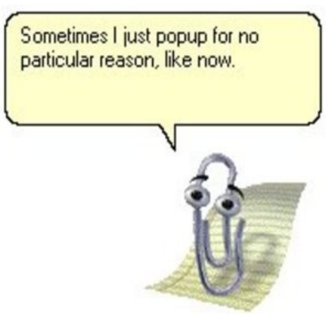
Ecole Centrale Paris (ECP)

March, 22, 2010

The Preference Elicitation Problem

- AI agents act on behalf of the user
- Specific user needs and preferences
  - Quality of these services are only as good as the user model
  - **BUT** user model is expensive to acquire
  - Only partial utility/preference information is available
  - Optimization with uncertain objective function

[Viappiani, AI Magazine 2008]



Preference-based Search

- Product configuration
- Large collection of outcomes
  - Users are not familiar with available items and features
  - Users do not know their preferences: theory of pr*eference construction [Payne]*
  - Biases in decision: framing, prominence, means-objectives *[Gilovich, Kahneman]*

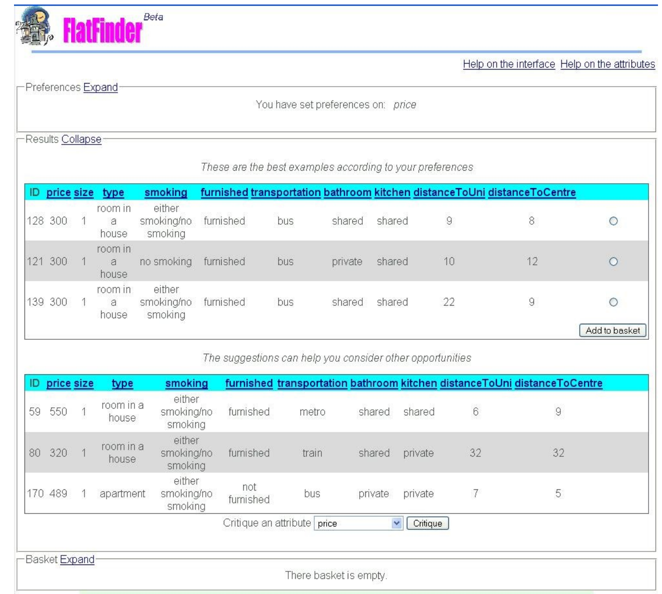
Biases in Decision Making

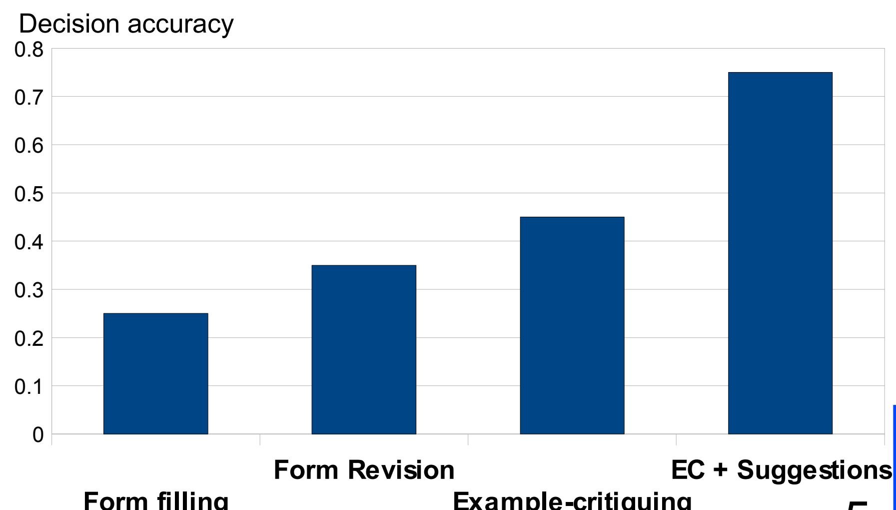
- Example: system asks about preferred airline first
  - Real preference about price:
    - *Believe* 'swiss' to be cheap → User answer 'Swiss is preferred'
    - Return an very expensive flight!
  - Airline=Swiss is a *means-objective*

Mixed initiative Interfaces

- Example-critiquing
  - Flexibility in product navigation
  - Prevent behavioral biases
- Adaptive Strategies of Suggestions

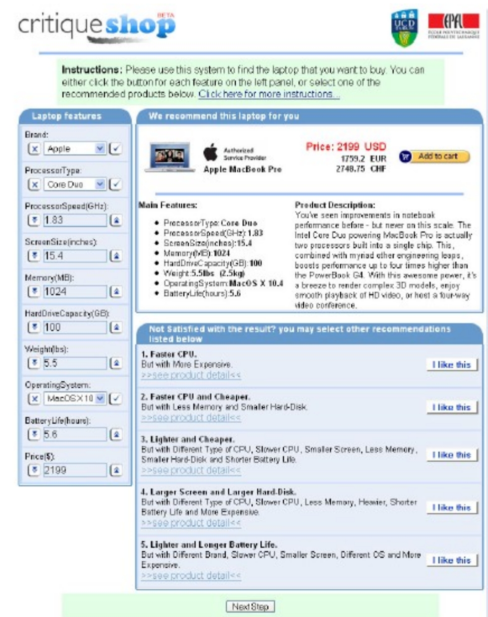
  [Viappiani, Faltings, Pu, JAIR 2006]
  - Bayesian learning
  - Stimulate expression of correct preference
  - Avoid framing
  - Evaluated with User Studies
  - Better preference model and increase decision accuracy



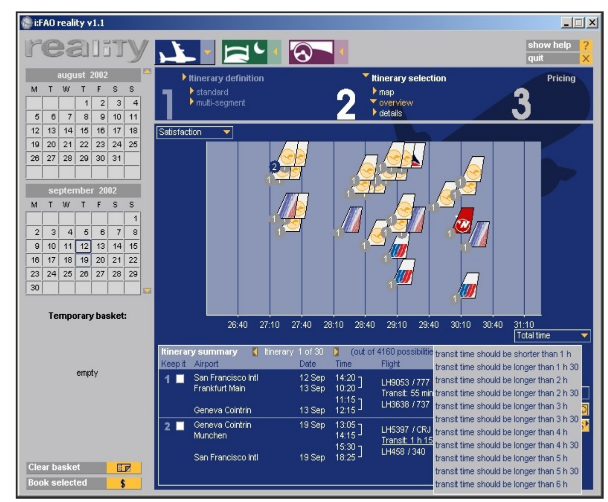


Critiquing Interfaces

- Despite all the advantages they...
  - Do not provide utility guarantees
  - User might be locked in a *local optimum*
  - Lack support for trade-offs



[Pu, Reilly, Smyth, et al.,EC 2008]



[McTorrens, Pu, Faltings, Comp Int. 2004]

Recommendations with an Explicit Utility Model

- Associate user's actions with a precise, sound semantics
  - E.g. critique impose linear constraints on a user utility function
- Advantages of our approach
  - Suggest a *set* of products
  - Bound the difference in quality of the recommendation and the optimal option of the user
  - Determine which options and critiques carry the most information
  - Suggest when *terminate* the process
- We adopt the notion of ***minimax regret*** to face utility uncertainty
  - Extend it to the case of a set of *joint* recommendations

Structure of the Talk

1. **Minimax Regret**
2. Optimal Recommendation Set
3. Feature Elicitation

Utility Model

- Finite set of *decisions* ***X***
  - Feasible set ***X*** defined by constraints, product DB, etc.
- *Utility representation critical to assessment*
- *u(x; w) parametrized compactly*
  - Vector w encodes the utility functions
  - Linear: $u(x; w) = w \cdot x$
  - Others: linear/additive, generalized additive models

Minimax Regret definition

W = set of feasible utilility parameters

X= set of products

x = recommendation

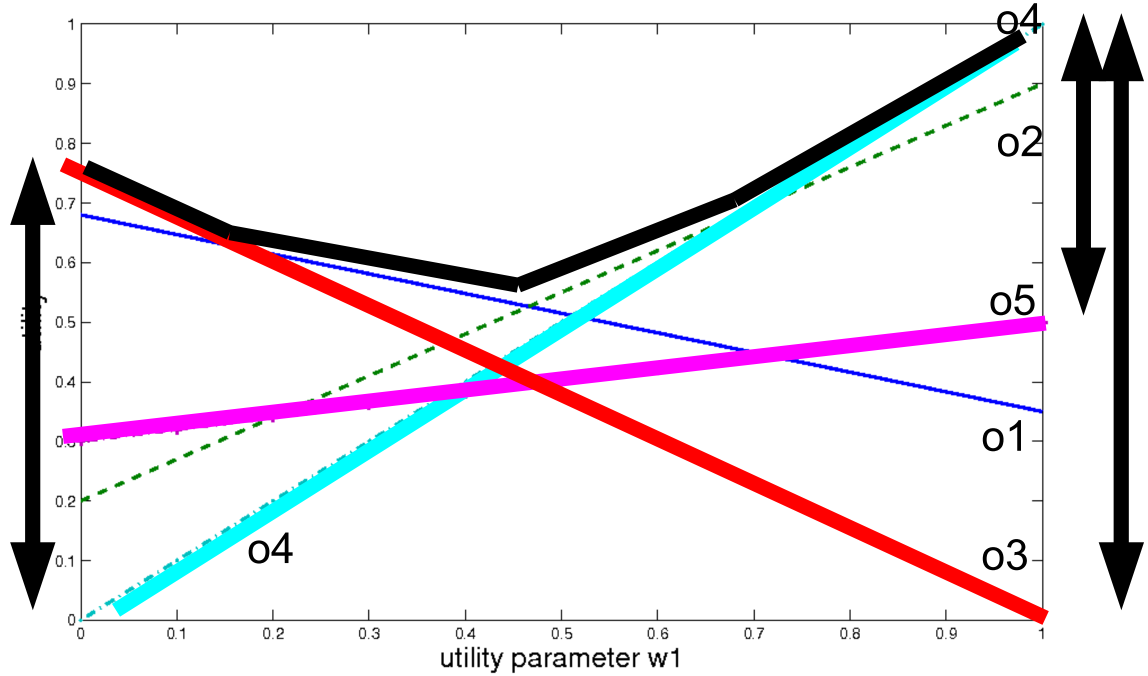
- Max regret

  $MR(x; W) = \max_{y \in X} \max_{w \in W} u(y; w) - u(x; w)$

- Minimax regret and minimax regret optimal x^*_W :

  $MMR(W) = \min_{x \in X} MR(x, W) \quad x^*_W = \arg\min_{x \in X} MR(x, W)$

	Feature 1	Feature 2
o_1	0.35	0.68
o_2	0.9	0.2
o_3	0	0.75
o_4	1	0
o_5	0.5	0.3



$U(x; w_1) = w_1 * f_1(x) + (1-w_1) * f_2(x)$

w_1 unknown

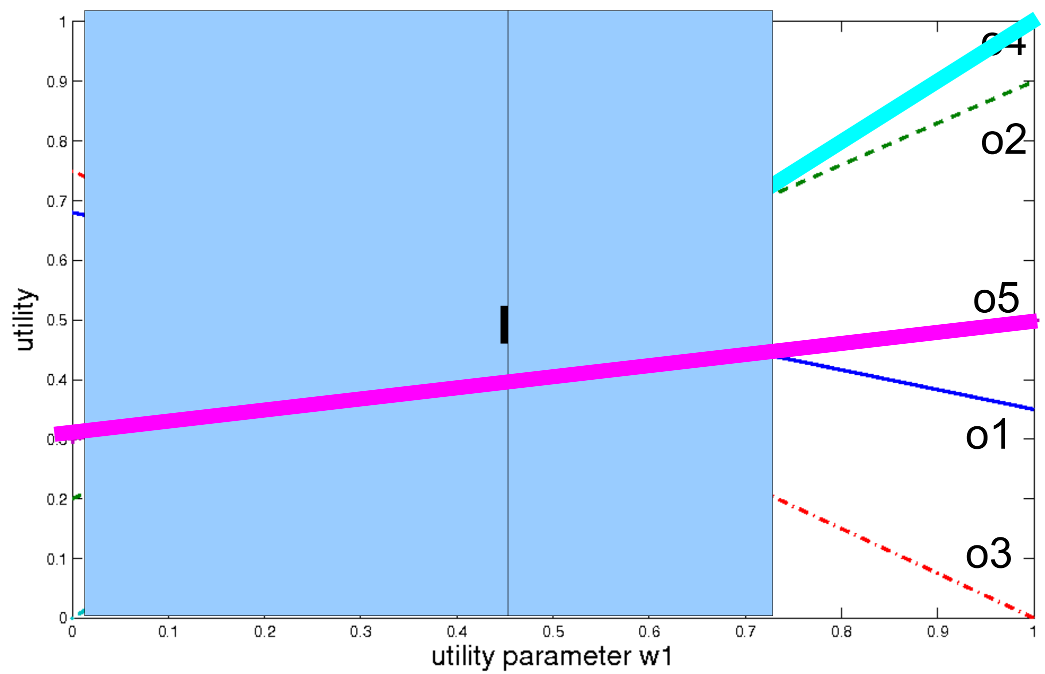
	Adversary	**MR**
o_1	o_4	0.65
o_2	o_3	0.55
o_3	o_4	1
o_4	o_3	0.75
o_5	o_4	0.5

o_5 minimax **regret optimal**

Regret-based recommender

W set of feasible utility functions

1) Initialize *W* with initial constraints
2) **DO** Generate current recommendations
3) Refine *W* given user's feedback
4) **LOOP** until user stops OR regret < Ɛ



Initial minimax regret = 0.5

User: o2 better than o1 → regret = 0.07

User: o4 better than o2 → regret = 0

Minimax Regret Computation

Minimax regret can be formulated as a MIP

- Benders' decomposition + constraint generation techniques

$$\min_{x} \max_{w} u(x^*_w; w) - u(x; w); \qquad x^*_w = \arg\max_{x \in X} u(x; w)$$

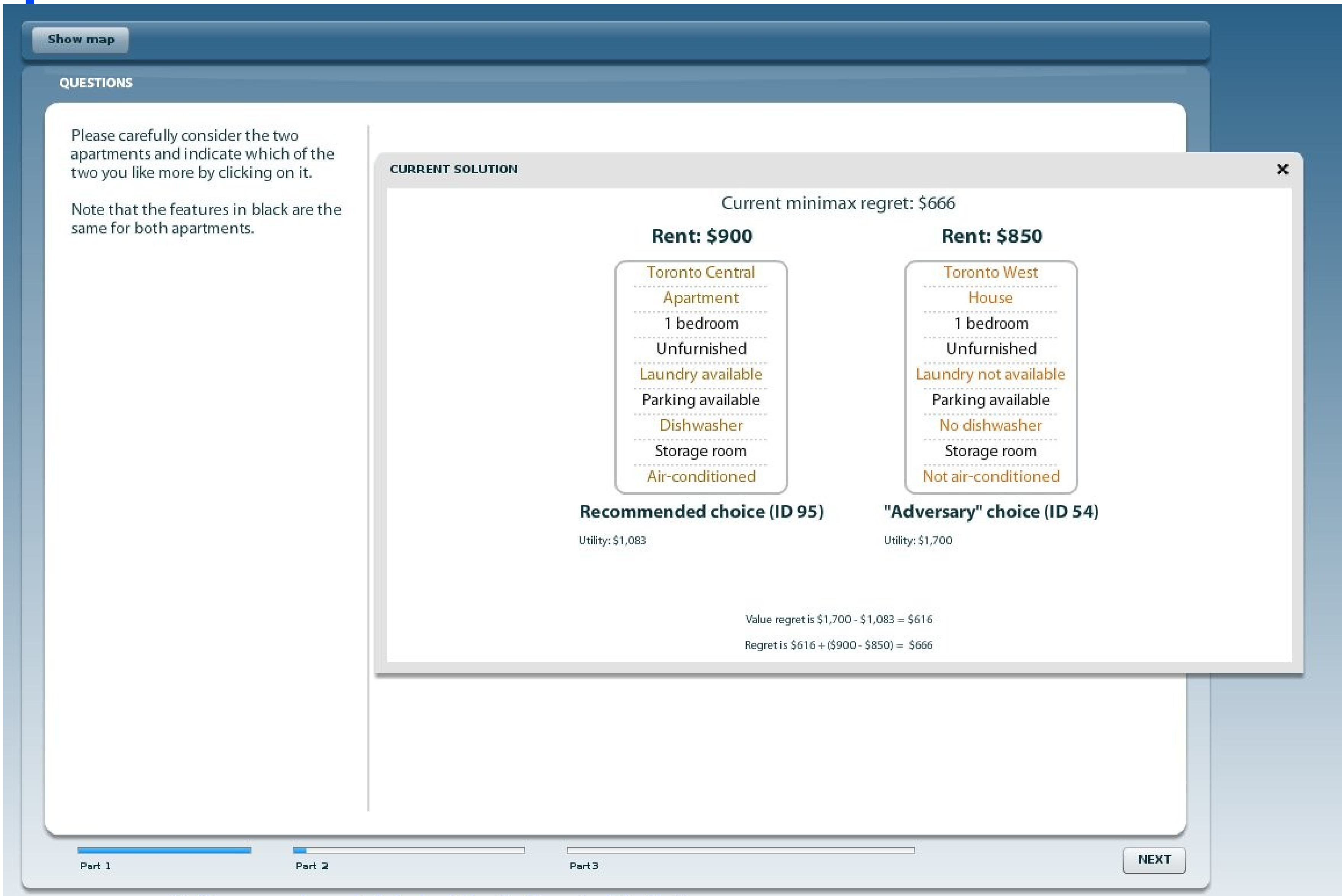
$$s.t. \quad R > u(x^*_w; w) - u(x; w) \qquad \forall w \in Vert(W)$$

$$\min_{x,R} \; R$$

$$s.t. \quad R \geq u(x^*_w; w) - u(x; w) \quad \forall w \in GEN \subset Vert(W)$$

Max Regret Computation

- Can be encoded as a MIP for a variety of utility models (additive, GAI) and configuration problems
- Can be computed with a sequence of LP for database problems



UTPREF [Braziunas]

Structure of the Talk

1. Minimax Regret
2. **Optimal Recommendation Set**
3. Feature Elicitation

[work with Craig Boutilier]

Recommendations Sets

- Show products that are both
  - *Expected* to be rated highly
  - *Maximally informative* should we have feedback
- This work: **optimal recommendation set** given a *sound decision-theoretic semantics* of the user interaction

Utility of a set

The value of a *set* is dependent on the elements of the set *jointly.*

We assume:

- $\text{Utility}\left(\begin{matrix} A \\ B \\ C \end{matrix}\right) = \max \left\{ \begin{matrix} U(A) \\ U(B) \\ U(C) \end{matrix} \right\}$
- A recommendation set gives “shortlisted” alternatives
- Reasonable in practice: apartment search example

Regret → Setwise Regret

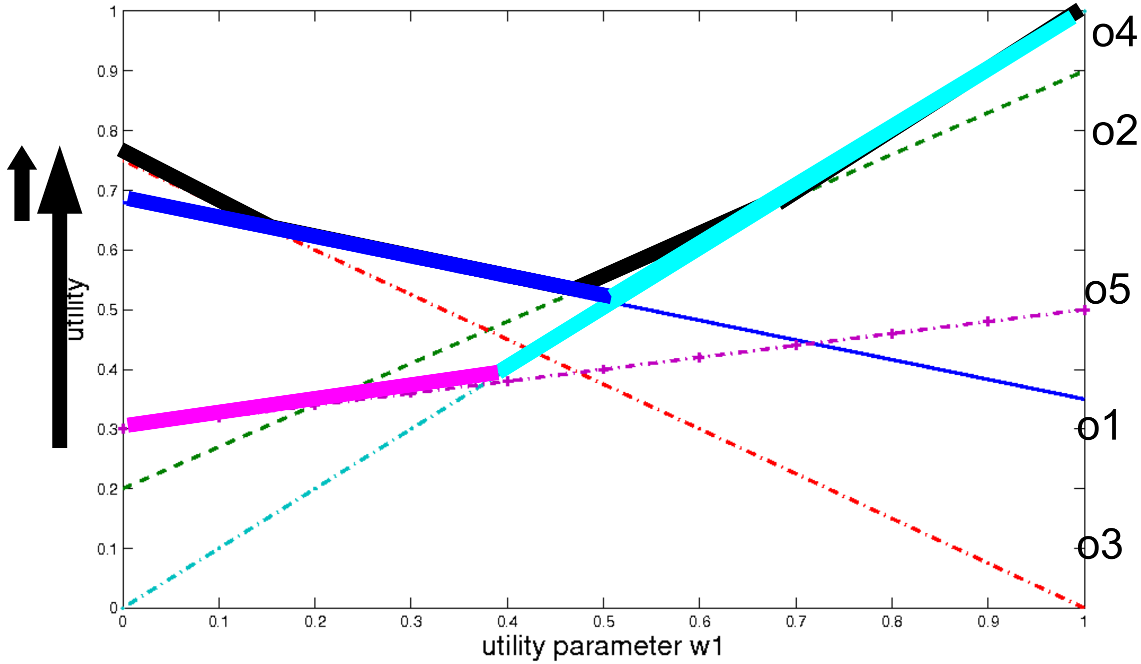
- We chooses the set of k options first, but *delay* the final choice from the slate after the adversary has chosen a utility function *w in W*
- Minimum difference btw options in the slate and (real) best option

- The setwise max regret SMR(Z; W) of a set Z:

$$SMR(Z; W) = \max_{y \in X} \max_{w \in W} \min_{x \in Z} u(y; w) - u(x; w)$$

- The *setwise* minimax regret SMMR(W) and the optimal set Z^*_W :

$$SMMR(W) = \min_{Z \subset X : |Z|=k} SMR(Z, W) \qquad Z^*_W = \operatorname{argmin}_{Z \subset X : |Z|=k} SMR(Z, W)$$

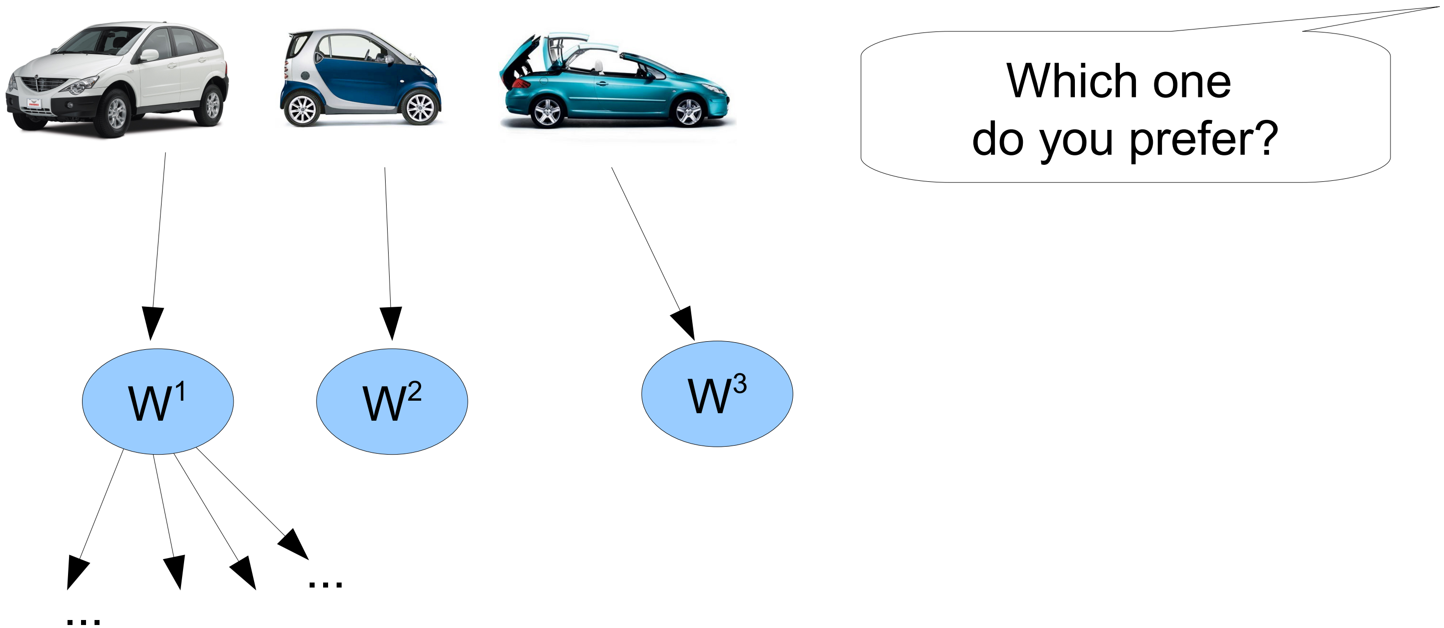


Set	Adversary	w_1	**SMR**
$\{o_1, o_4\}$	o_3	0	0.07
$\{o_1, o_2\}$	o_3	1	0.1
$\{o_3, o_2\}$	o_4	1	0.1
$\{o_3, o_4\}$	o_3	0.42	0.11
$\{o_5, o_4\}$	o_4	0	0.45

$\{o_1, o_4\}$ *setwise* minimax **regret optimal**

Incorporating User Feedback

Slate **Z** of k options viewed as a "query set" - user picks one



- Worst-case Regret (wrt each possible answer)
  - **WR**(Z) = max [MMR(W^1), MMR(W^2), ..]

Two objectives

- *Minimize SMR*: recommendation set with lowest loss
- *Minimize WR*: query set with greater regret reduction after user answer (wrt worst-case)

- Straight minimization of WR is hard

- Relation between SMR and WR ?

Theorem

- The optimal recommmendation set Z^*_W is also the (myopically) optimal query set wrt worst-case regret (WR)

→ ***"Best recommendation set = best query set"***

- The optimal query set can be chosen without enumeration
  - We can compute setwise regret efficiently
  - *Setwise minimax regret* can be formulated as a MIP
  - Benders' decomposition + constraint generation techniques
  - Approximation techniques

Hillclimbing procedure
“minimax-regret rewriting”

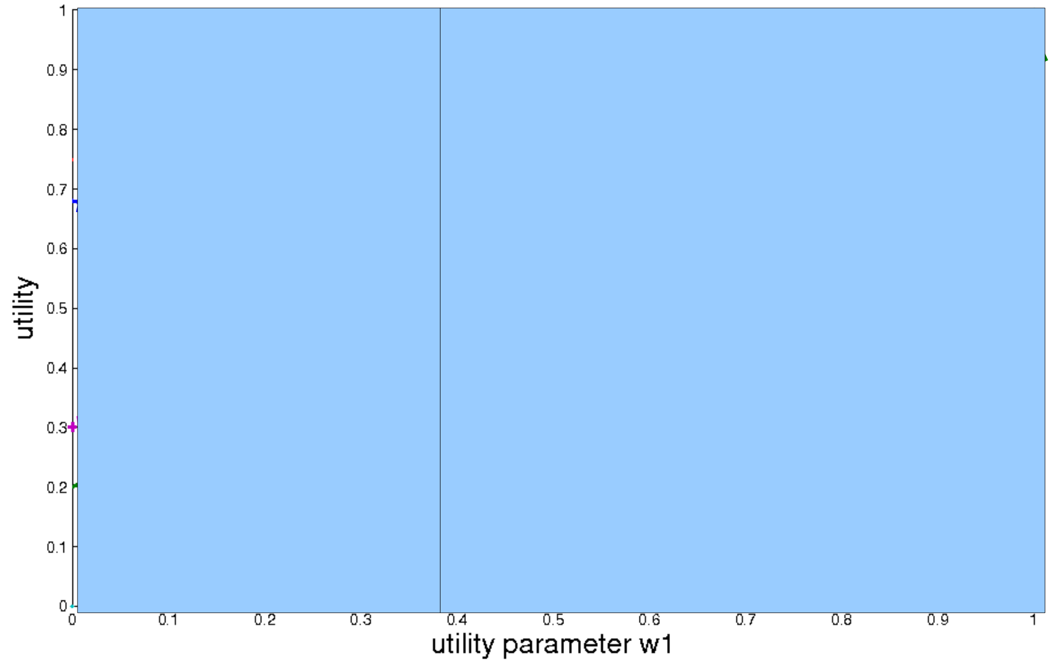
Given a set $Z = \{x^1,..,x^k\}$

DO

- Partition the utility space
- X^1 option preferred → new space $W^{Z\to 1}$
- ...
- X^k option preferred → new space $W^{Z\to k}$
- Replace x^i with $x^*_{W^i}$, the MMR-optimal in W^i

■ **WHILE** SMR(Z^{new}) < SMR(Z)

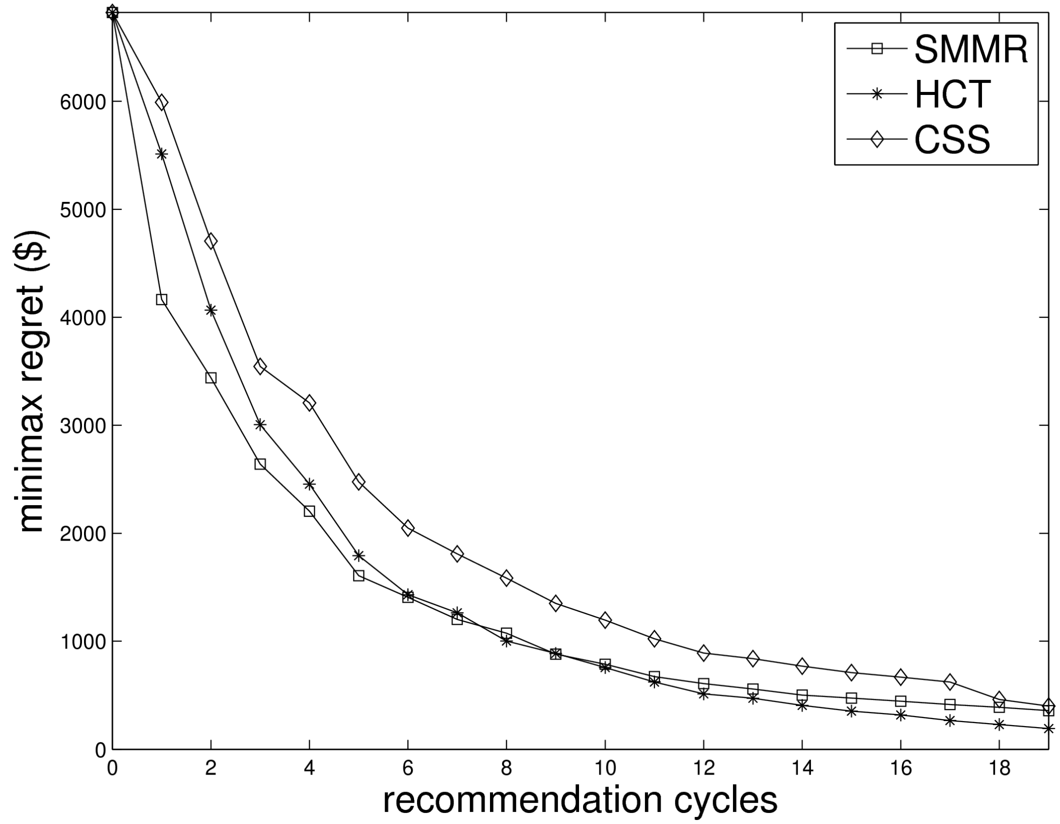
The inner replacement can be proved not to increase SMR



- Start with $\{o_5, o_4\}$
- Assume o_4 better than o_5
  - Compute MMR: this gives o_2
- Assume o_5 better than o_4
  - Compute MMR: this gives o_1
- New query $\{o_1, o_2\}$

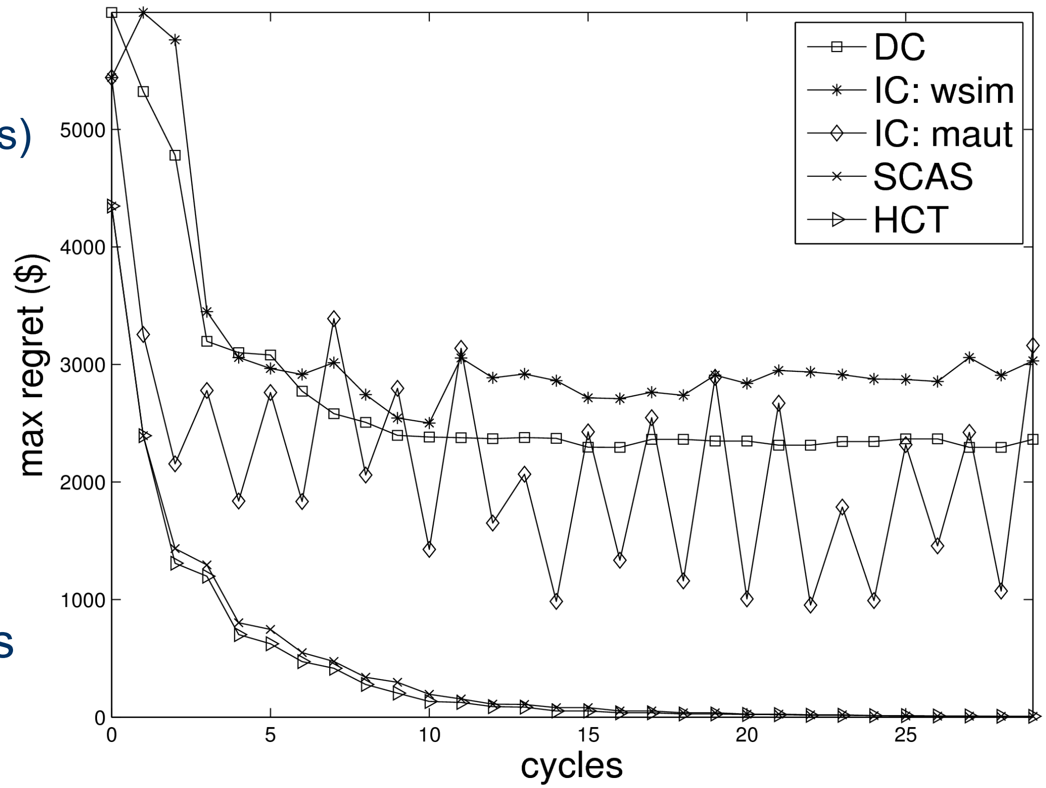
Empirical Results

- Randomly generated *quasilinear* utility functions
- Real dataset (~200 options)
- User iteratively picks preferred option in a pair (k=2)
- Measure regret reduction
- SMMR recommendations are significantly better than CSS
- Hillclimbing (HCT) is as good as SMMR



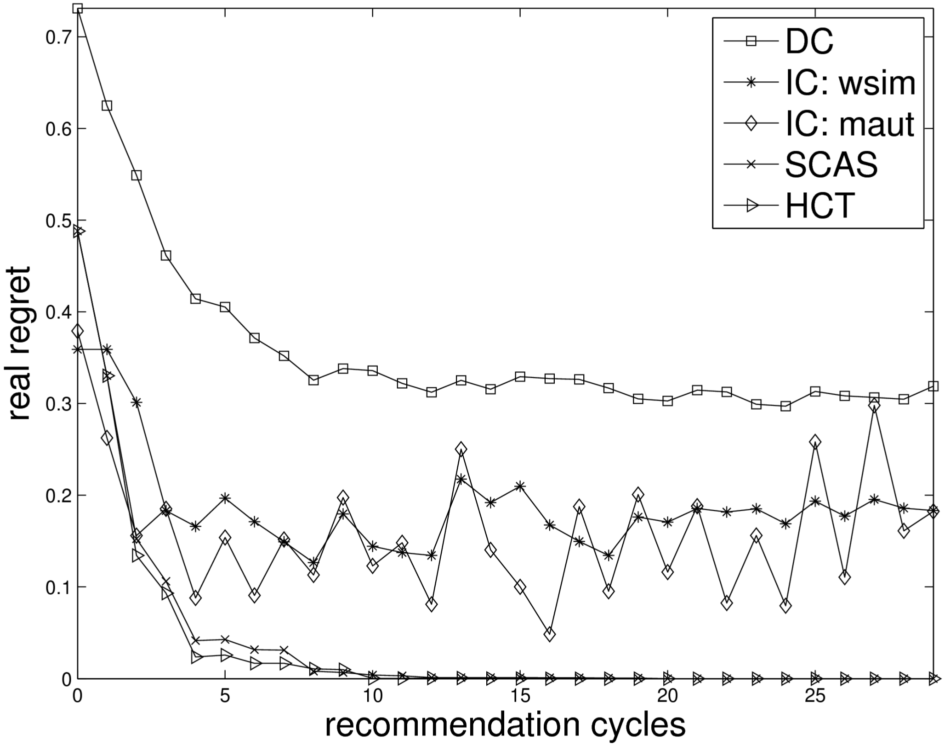
Critiquing Simulation

- Simulate a critiquing session
  - Quasilinear utility model
  - Synthetic dataset (5000 options)
- "Optimizing" user chooses best critique wrt real utility
- Alternate btw
  - Selection of feature to improve ('unit critique')
  - Selection among a set of 3 suggestions
- HCT-based set recommendations gives best regret reduction



Real Loss

- Real loss (regret) is the difference to the actual optimum
- Set size k=3
- Regret-based recommender give optimal recommendation in very few cycles



Structure of the Talk

1. Minimax Regret
2. Optimal Recommendation Set
3. **Feature Elicitation**

[work with Craig Boutilier and Kevin Regan]

Subjective Features

- Preference elicitation usually focuses on “catalog” attributes (or product specifications)
  - *Engine, size, color, fuel economy; number of bedrooms,…*
- We consider “user-defined” *subjective* features
  - Constructed on the fly
  - Application to critiquing interfaces (eg Findme)
  - User can focus on *fundamental objectives* [Keeney]

Subjective Features

- SAFE CAR

CrashTestRatings > Good **AND** easy to park

HasSideAirbags **AND** isSuv

size=big **AND** brand=volvo

Feature Elicitation

- Feature elicitation vs. classical concept learning
  - *Learn just enough* about a concept in order to make a *good decision*
  - Reward model + feasibility constraints → near optimal recommendation with weak concept knowledge
  - Minimize user queries

Example: preference for ***sporty*** cars, BUT luggage capacity more important.

If all “sporty cars” have small luggage capacity, it is not worth continuing to learn more about *sporty!*

Abstract Model for Feature Elicitation

- Product space $\boldsymbol{X} \subseteq Dom\{X_1 \ldots X_n\}$
  - Reward $r(\boldsymbol{X})$ reflects utility for catalog features
  - Concept $c(\boldsymbol{X})$ drawn from some hypothesis space H
  - Bonus p: additional utility for an $\boldsymbol{x}$ satisfying $c(\boldsymbol{x})$
  - Utility $u(\boldsymbol{x}; \boldsymbol{c}) = r(\boldsymbol{x}) + p\ c(\boldsymbol{x})$
  - Goal: recommend products with highest utility
- Version space **V**
  - Subset of H that is consistent with the current knowledge about the concept

Minimax Regret over Concepts

- Let $V \subseteq H$ be current version space
  - $c \in V$ iff c respects prior knowledge, responses, etc.
- The adversary chooses concept and witness $\mathbf{x}^w$

$$MR(\mathbf{x}; V) = \max_{c \in V} \max_{\mathbf{x}' \in \mathbf{X}} u(\mathbf{x}'; c) - u(\mathbf{x}; c)$$

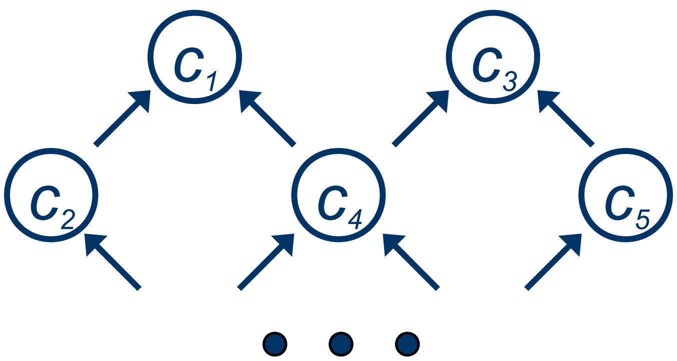
$$MMR(V) = \min_{\mathbf{x} \in \mathbf{X}} MR(\mathbf{x}, V)$$

$$\mathbf{x}^*_V = \arg\min_{\mathbf{x} \in \mathbf{X}} MR(\mathbf{x}, V)$$

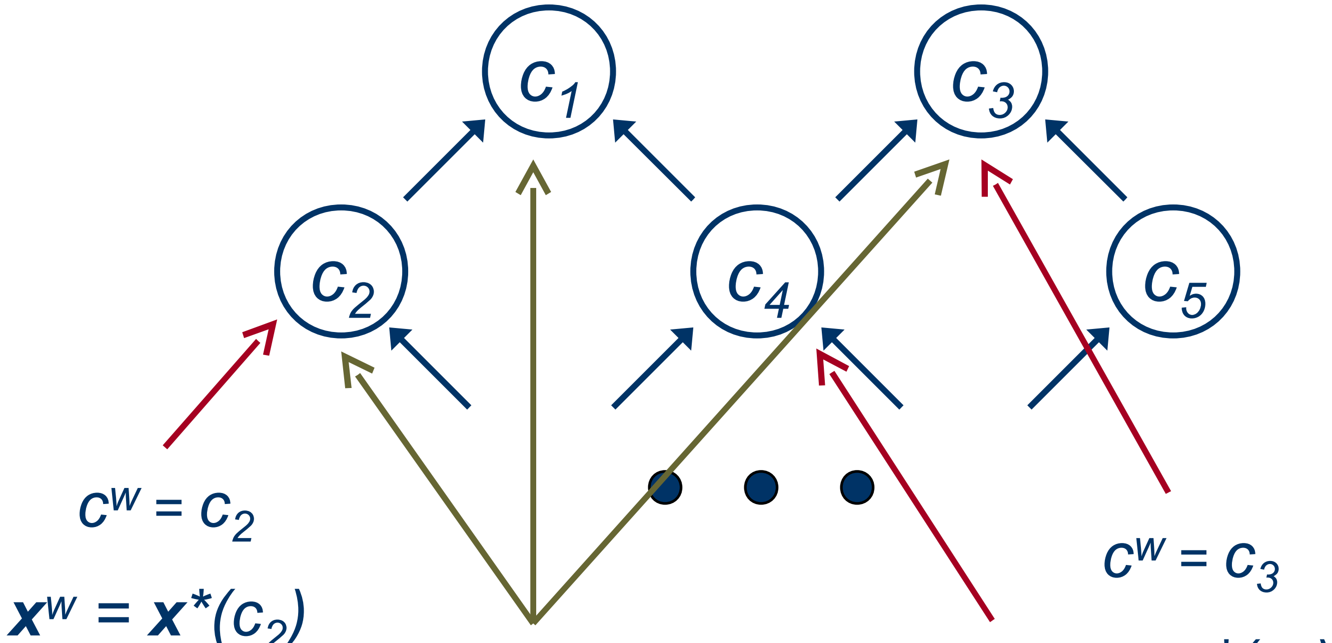
- If $MMR(V) = \varepsilon$, $\boldsymbol{x}^*$ is ε–optimal.

Characterizing MMR-Optimal Soln

- MMR-optimal soln $\boldsymbol{x}^*$, $\boldsymbol{x}^w$, c^w: interesting structure
  - General-specific lattice $>$ over V: $c > c'$ iff $c' \subseteq c$
  - Best $\boldsymbol{x}$ satisfying c: $r^*(c) = max \{ r(\boldsymbol{x}) : \boldsymbol{x} \in c, \boldsymbol{x} \in \boldsymbol{X} \}$
  - Induces reward-ordering over V: $r^*(c_1) > r^*(c_2) > \ldots$
  - Reward ordering respects GS ordering



Characterizing MMR-Optimal Soln



PROPOSITION
x* is either the product with highest reward *OR*

1) There is a concept in the version space V that satisfies it
2) x* in argmax{ r(**x**) : **x** in $C_1 \cap .. \cap C_i$ } for some i≥1
3) either c^w in C_1, or c^w in C_{i+1}

Computing MMR: Conjunctions

- MMR encoded MIP
  - Details and formulation depend on the hypothesis space
  - Various encoding tricks to encode concept satisfaction
  - Special case: conjunctions, memberships queries
    - *e.g., “Do you consider this to be a safe car?”*

$$
\begin{aligned}
\min \quad & \delta \\
\text{s.t.} \quad & \delta \geq r(\mathbf{x}_c) - r(X_1, \cdots X_n) + p(\mathbf{x}_c, c) - pI^c \quad \forall c \in V \\
& I^c \leq X_j \quad \forall c \in V, \forall x_j \in c \\
& I^c \leq 1 - X_j \quad \forall c \in V, \forall \overline{x}_j \in c
\end{aligned}
$$

$$
\mathbf{x}_c = \arg\max_{\mathbf{x} \in \mathbf{X}} u(\mathbf{x}; c)
$$

Computing MMR: Conjunctions

- Maximization sub-problem
  - Find maximally violated constraint: concept that maximizes regret *MR(**x***,V)*
- Let E^+, E^- be positive, negative instances

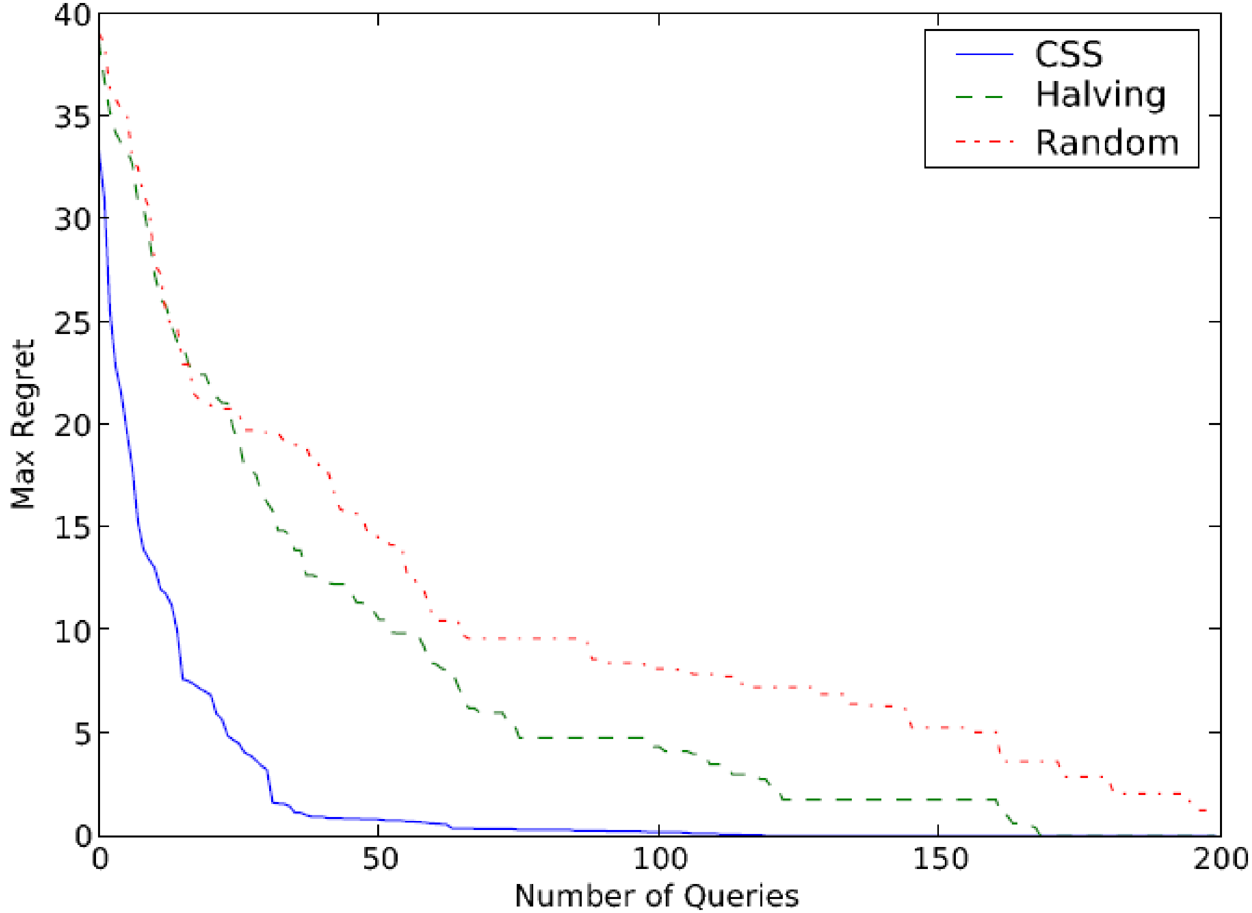
$$
\begin{aligned}
\max \quad & r(X_1, \cdots, X_n) - r(\mathbf{x}) + pB^w - pB^x \\
\text{s.t.} \quad & B^w + I(x_j) \leq X_j + 1.5 \quad \forall j \leq n \\
& B^w + I(\overline{x}_j) \leq (1 - X_j) + 1.5 \quad \forall j \leq n \\
& B^x \geq 1 - \sum_{j:\mathbf{x}[j] \text{ positive}} I(\overline{x}_j) - \sum_{j:\mathbf{x}[j] \text{ negative}} I(x_j) \\
& \sum_j I(\neg \mathbf{y}[j]) = 0 \quad \forall \mathbf{y} \in E^+ \\
& \sum_j I(\neg \mathbf{y}[j]) \geq 1 \quad \forall \mathbf{y} \in E^- \\
& (X_1, \cdots, X_n) \in \mathbf{X}
\end{aligned}
$$

Query Strategies

- Aim: reduce regret quickly
- Several strategies using membership queries:
  1. *Halving*: aims to learn concept directly
     - “random” query $\boldsymbol{x}$ until positive response; then refine (unique) most specific concept in V (negate one literal at a time)
  2. *Current Solution (CS)*: tackle regret directly
     - If $\boldsymbol{x}^*$, $\boldsymbol{x}^w$ both in $c^w \rightarrow$ query $\boldsymbol{x}^w$ (unless certain)
     - If $\boldsymbol{x}^w$ in c^w but not $\boldsymbol{x}^* \rightarrow$ query $\boldsymbol{x}^*$ (unless certain)
     - If $\boldsymbol{x}^*$, $\boldsymbol{x}^w$ both not in $c^w \rightarrow$ query $\boldsymbol{x}^w$ if $\boldsymbol{x}^w$ (unless certain)
  3. Several variants show modest improvements

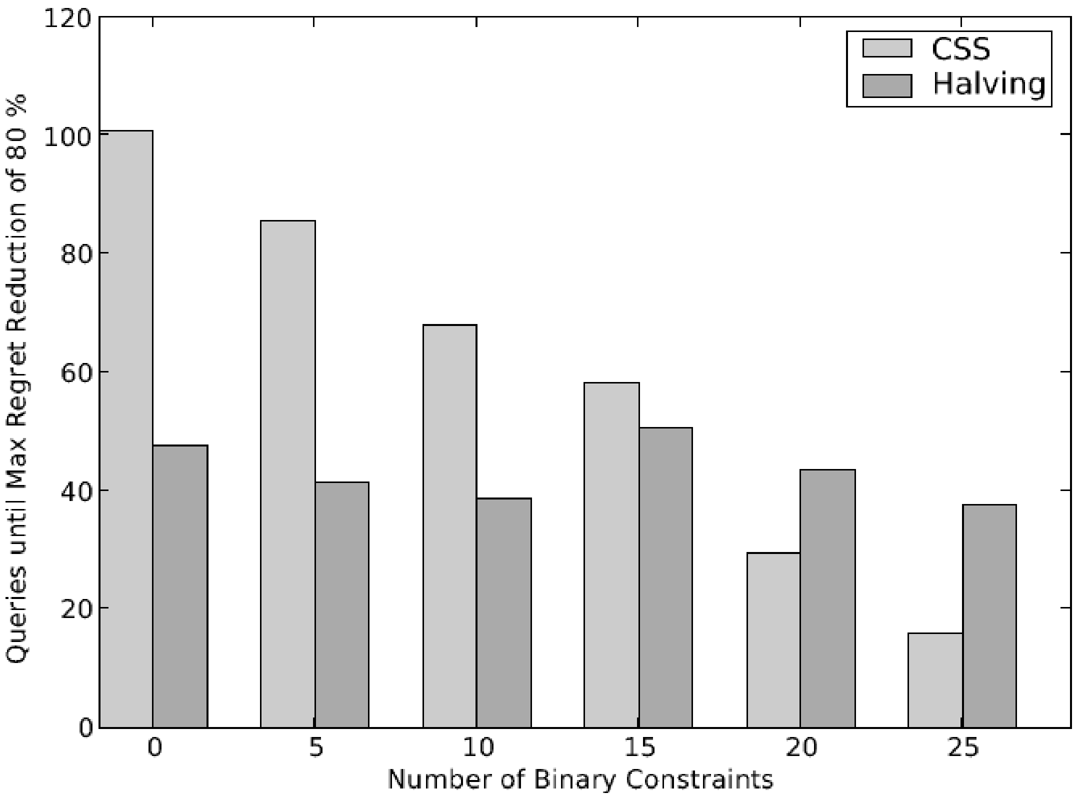
Experimental Results

- 30 variables, 20 random binary constraints, concepts have size 10, random reward/bonus, bonus = 25% of max reward



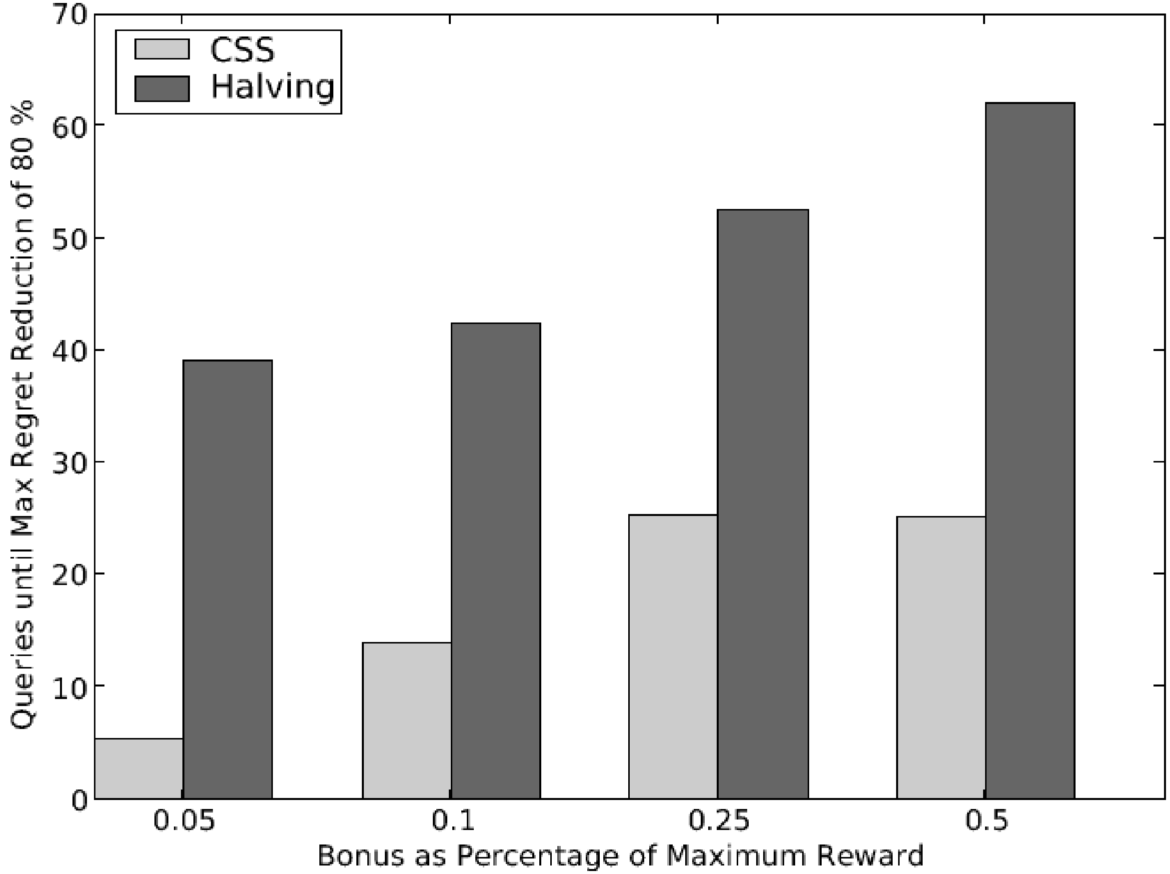
Varying Constraint Tightness

- Tighter constraints: sparser solution sets, more variability in r^* values, more concepts in V without positive instances in $\boldsymbol{X}$
  - shown: number of queries to reach regret reduction of 80%



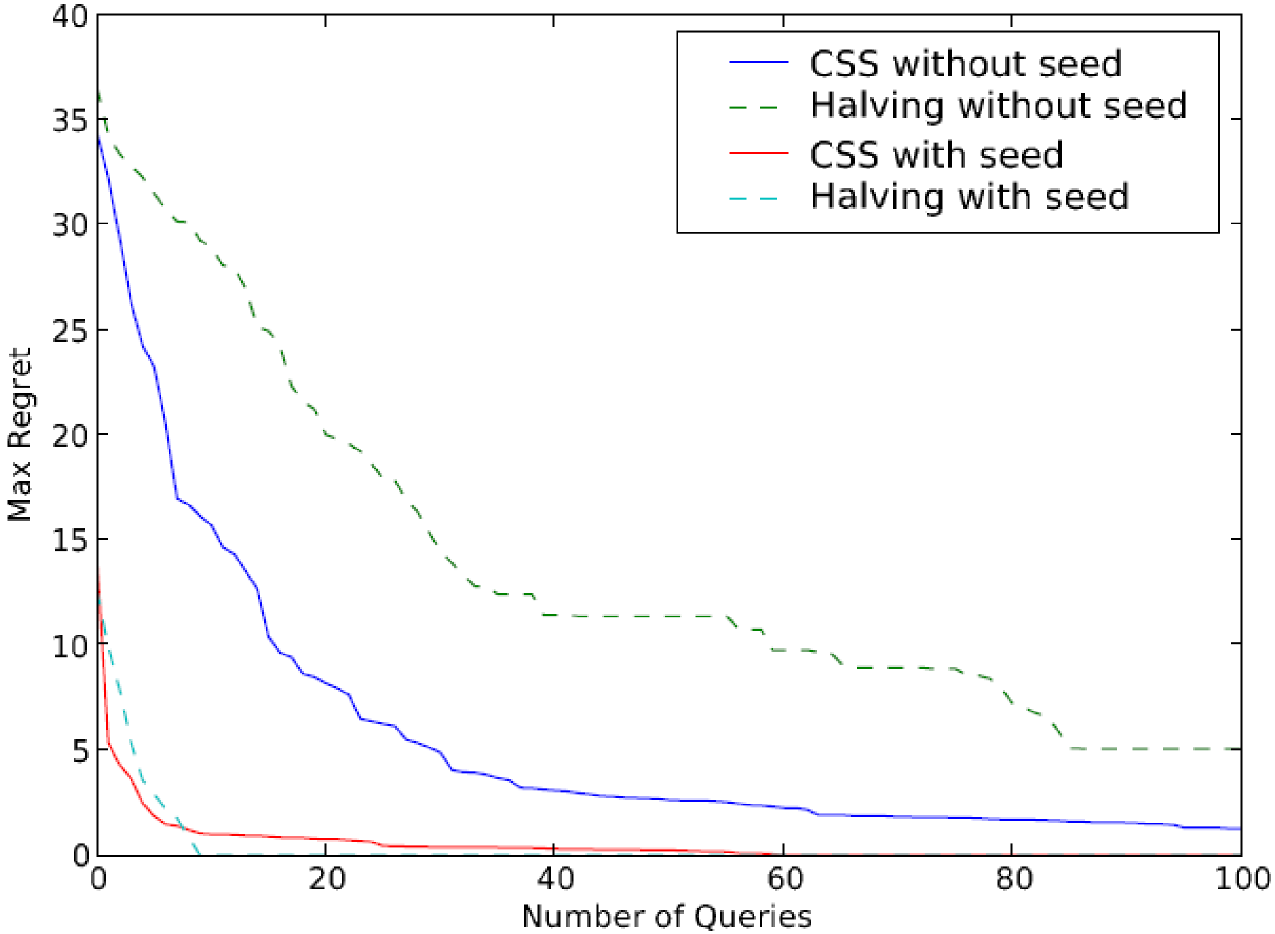
Varying Relative Bonus

- Greater bonus value: refining the concept becomes more critical
  - shown: queries to reach regret reduction of 80% (20 constraints)



Positive Instance as Seed

- Once positive instance found, true "halving" kicks in
  - assume user identifies a positive example immediately



Feature + Utility Uncertainty

- Utility **u(x; w, c)** function of *unknown* weights and feature
  - Require *simultaneous utility and feature elicitation*
  - Doing one "completely" followed by other is wasteful
- Query strategies: which type of query?
  - ***Interleaved* strategies (*I*)** asks membership query when 'reward' component of regret is higher
  - ***Phased Strategies (Ph)*:** always ask membership when uncertain about concept
  - ***Combined comparison-membership query* (CCM):** asks both comparison and membership queries about x* and x^w
    - In general, counts as 3 queries
- Query strategies: what to ask?
  - Comparison query: CCSS
  - Membership query: MCSS vs Halving

Comparison Queries in Joint Model

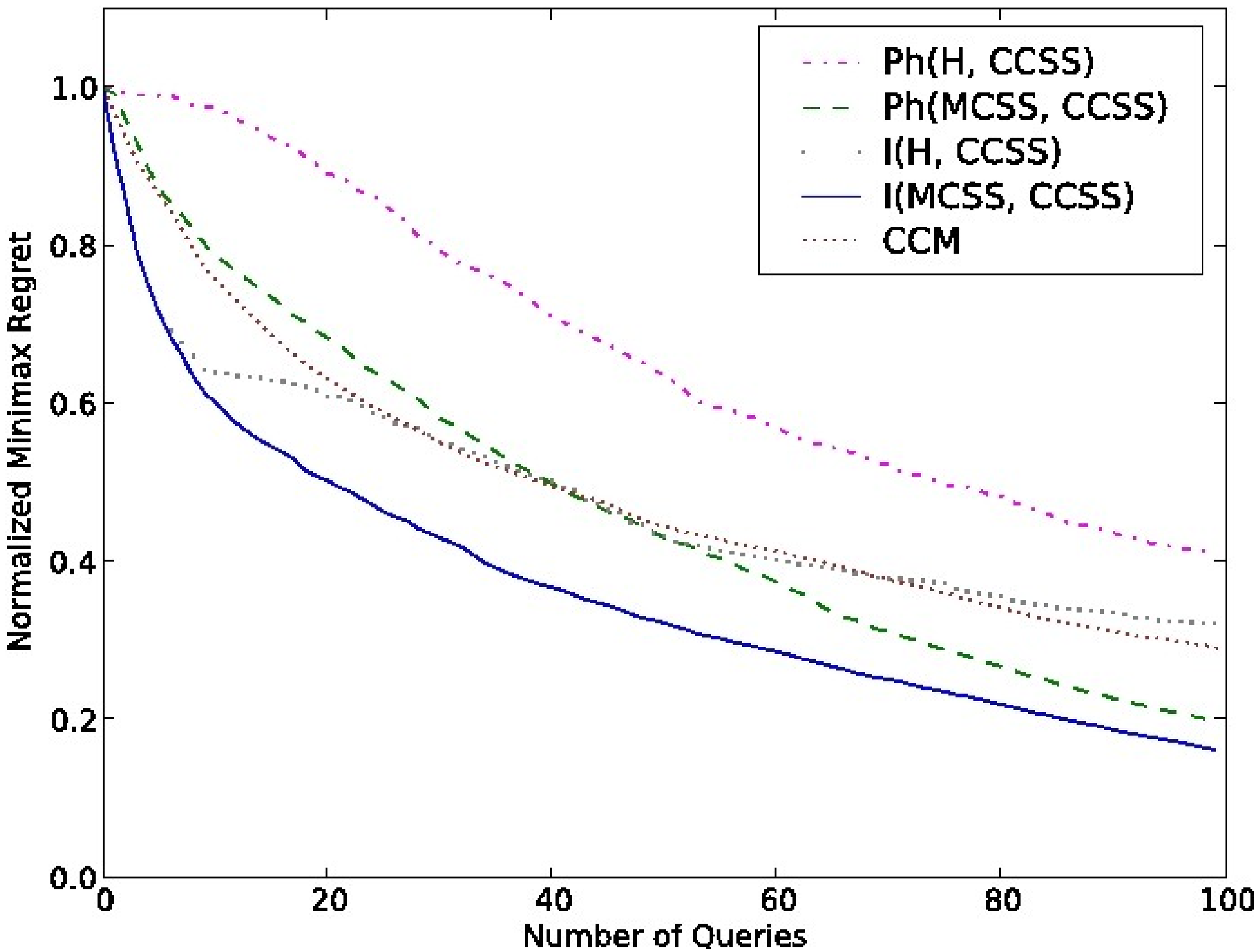
- User prefers ***x*** to ***y***
  - no feature uncertainty: linear constraint *w**x*** > *w**y***
  - feature uncertainty, more complicated
    - ask membership: linear; e.g., *w**x*** + *p* > *w**y***
    - *unknown* membership: conditional constraints
      - *can be linearized in MIP*

$$w\mathbf{x} - w\mathbf{y} > 0 \;\text{ if }\; c(\mathbf{x}), c(\mathbf{y})$$

$$w\mathbf{x} + p - w\mathbf{y} > 0 \;\text{ if }\; c(\mathbf{x}), \neg c(\mathbf{y})$$

$$w\mathbf{x} - w\mathbf{y} - p > 0 \;\text{ if }\; \neg c(\mathbf{x}), c(\mathbf{y})$$

$$w\mathbf{x} - w\mathbf{y} > 0 \;\text{ if }\; \neg c(\mathbf{x}), \neg c(\mathbf{y})$$



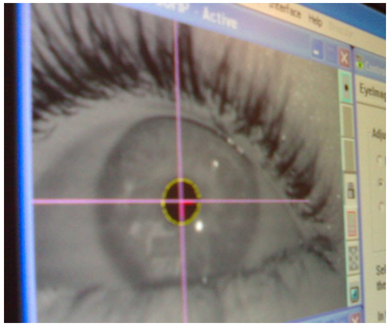
Interleaved elicitation strategies are better off than *phased* strategies (large problem size, 30 attributes)

Summary

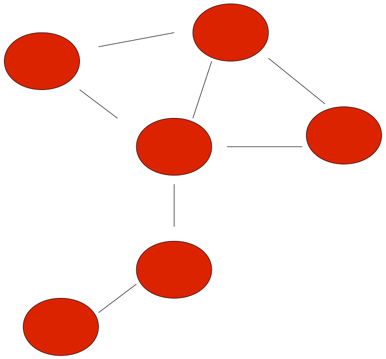
- Minimax regret optimization for recommendation and utility elicitation
- Formalization of recommendations of a joint set of alternatives
  - We proposed a new criterion *setwise regret*
    - Intuitive extension of regret criterion
    - Guarantee on the quality of the recommendation set
    - Efficient driver for further elicitation
- Optimal recommendations sets = optimal query sets
  - Computation & heuristics
- Subjective Features
  - Minimax regret formulation over concept
  - Interleaved elicitation of utility and features

Future works

- *Contextual* preference assessment
  - Technologies like *eye/gaze tracking* (much less expensive than in the past!)
  - Challenges:
    - Effective learning algorithms to identify user context, and
    - Provide situated responses



- Social Networks
  - Optimize diffusion
  - Leverage similarities between nodes



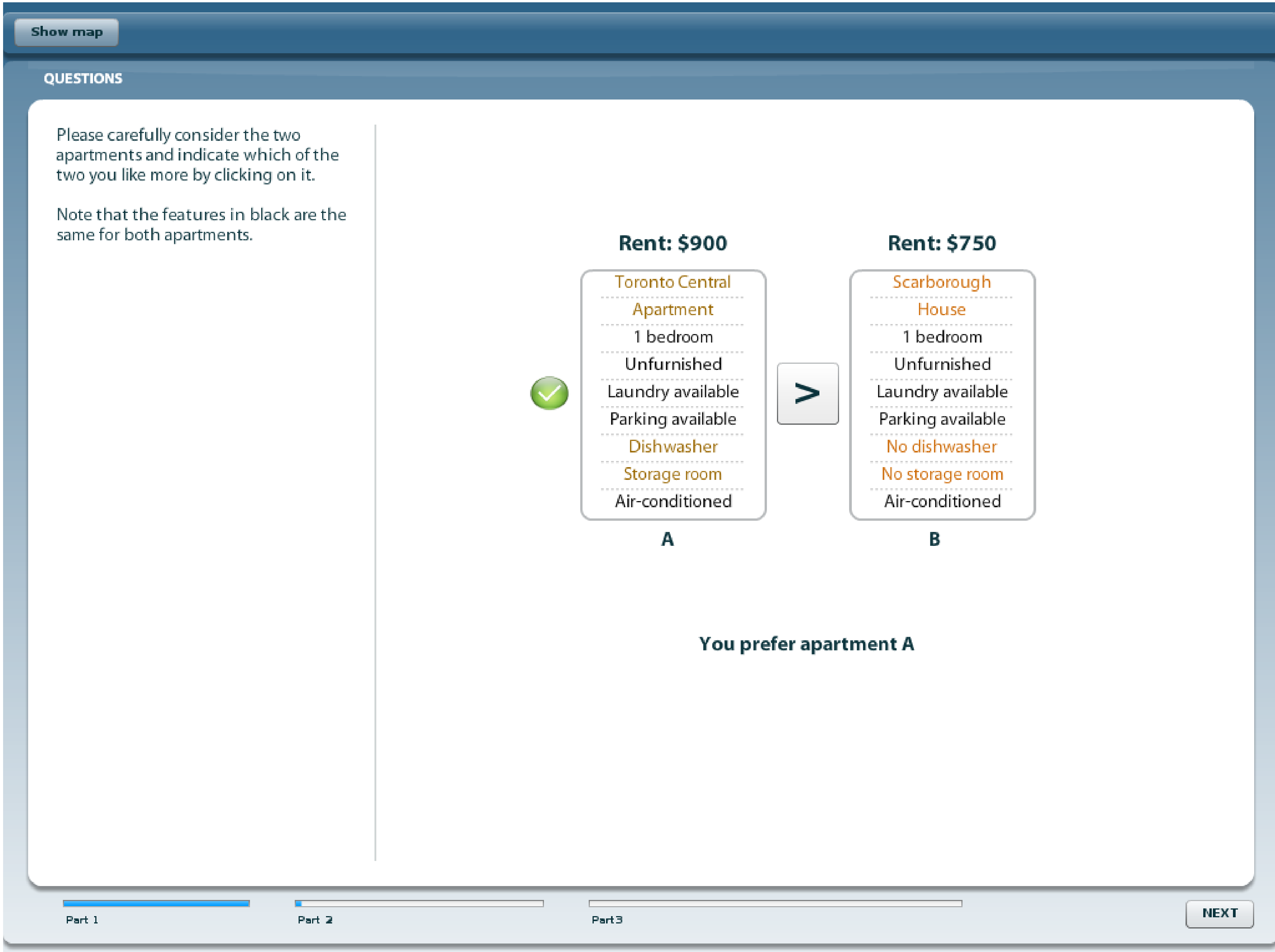
- Bayesian Approaches
- Noisy models
- Applications: ranking, computational advertisement, planning systems

Why Minimax Regret?

- Minimizes regret in presence of adversary
  - provides bound worst-case loss (cf. maximin)
  - *robustness* in the face of utility function uncertainty
  - We extend it to *concept uncertainty*
- In contrast to Bayesian methods:
  - useful when priors not readily available
  - can be more tractable; see **[CKP00/02, Bou02]**
  - user unwilling to “leave money on the table” **[BSS04]**
  - preference aggregation settings **[BSS04]**
  - effective elicitation even *if* priors available **[WB03]**

Constraint Generation

- Constraint generation: avoid enumeration of *V*
  - ***REPEAT***
  - Solve minimization problem with a subset *GEN* of *V*
    - The adversary's hands are tied to choose a couple *(w, y)* from this subset
    - LB of minimax regret
  - Find *max violated constraint computing MR(x)*
    - UB of minimax regret
  - Add the adversarial choice to *GEN*
  - Terminate **WHEN** UB = LB



Show map
QUESTIONS
Please carefully consider the two apartments and indicate which of the two you like more by clicking on it.
Note that the features in black are the same for both apartments.
Rent: $900
Toronto Central
Apartment
1 bedroom
Unfurnished
Laundry available
Parking available
Dishwasher
Storage room
Air-conditioned
A
>
Rent: $750
Scarborough
House
1 bedroom
Unfurnished
Laundry available
Parking available
No dishwasher
No storage room
Air-conditioned
B
You prefer apartment A
Part 1
Part 2
Part 3
NEXT

QUESTIONS

You are asked to decide whether the apartment on the left is "closer" in value to the TOP apartment or the BOTTOM apartment.

Features that are not shown (including price) **are the same for all three apartments**. Note that any features shown in grey are also the same for all apartments.

You have previously indicated that BOTTOM has the worst combination of features, and TOP has the best combination of features. On the scale from 0 to 100 (shown on the right of the bins) BOTTOM is at 0, and TOP is at 100. You should consider of **where** the apartment in question falls on this scale. If its value is between 0 and the tip of the slider, please drag it to the bottom bin; otherwise, drag it to the top bin.

House
1 bedroom
Toronto Central

TOP

Apartment
1 bedroom
Toronto Central

100
90
80
70
60
50
40
30
20
10
0

Basement
3 bedrooms
Toronto Central

BOTTOM

Part 1 | Part 2 | Part 3

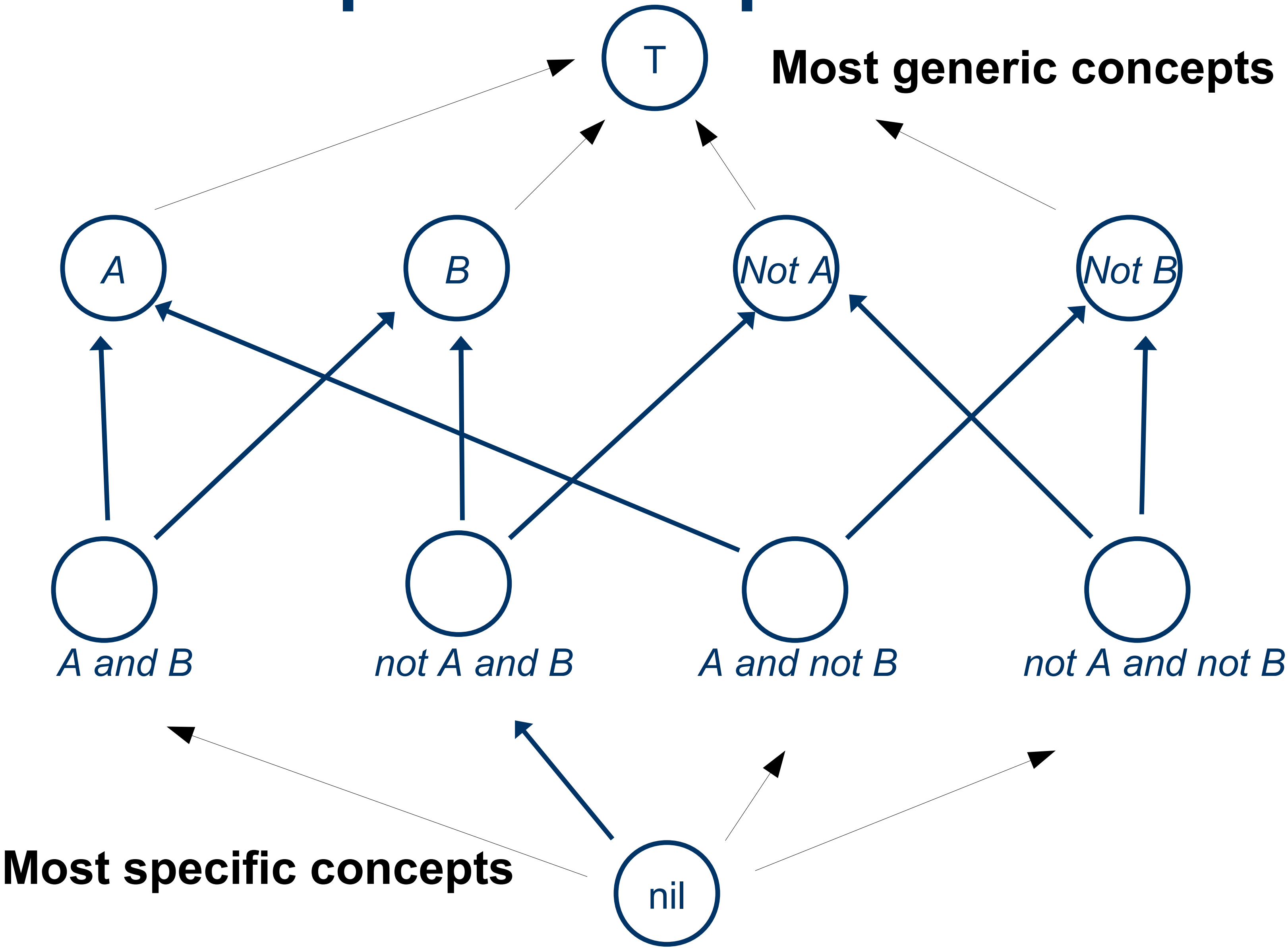
NEXT

Setwise Regret Computation

- Setwise minimax regret can be formulated as a MIP
  - Benders' decomposition + constraint generation techniques

$$
\begin{aligned}
\min_{M, I_w^j, \mathbf{X}^j, V_w^j} \quad & M \\
\text{s.t. } M \geq \sum_{1 \leq j \leq k} V_{\mathbf{w}}^j \quad & \forall \mathbf{w} \in \mathit{Vert} \\
V_w^j \geq \mathbf{w} \cdot (\mathbf{x}_{\mathbf{w}}^* - \mathbf{X}^j) + (I_w^j - 1) m_{big} \\
\forall j \in [1, k] \wedge \quad & \forall \mathbf{w} \in \mathit{Vert} \\
\sum_{1 \leq j \leq k} I_{\mathbf{w}}^j = 1 \quad & \forall \mathbf{w} \in \mathit{Vert} \\
I_{\mathbf{w}}^j \in \{0, 1\} \\
V_{\mathbf{w}}^j \geq 0 \quad \forall j \in [1, k], \forall \mathbf{w} \in \mathit{Vert}
\end{aligned}
$$

Version Space Example



(Single Item) Minimax Regret Computation

- Configuration problems
  - Benders' decomposition and constraint generation to break minimax program
- Discrete datasets
  - Adversarial search with two plys
  - Heuristics:
  - order to maximize pruning
    - Sample hypercube vectors