Retrieving the Structure of Utility Graphs Used In Multi-Item Negotiation Through Collaborative Filtering

Valentin Robu, Han La Poutré
CWI, Center for Mathematics & Computer Science
Amsterdam, The Netherlands

Multi-issue (multi-item) negotiation models

- Alternating offer game
- Indirect revelation, i.e. utility functions are not directly revealed
- Non zero-sum: reach an agreement close to Pareto-optimality

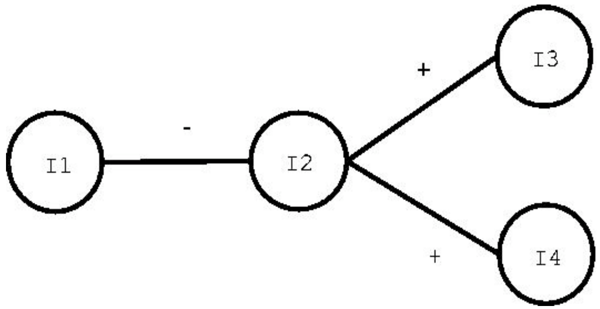
Utility function types used in negotiation:

- **Linearly additive:** very widely used in literature on bilateral bargaining
- **K-additive (e.g. for k=2):** $U_B = \sum_{i \in S} w_i I_i + \sum_{i,j \in S} w_{i,j} I_i I_j$

  - Fully expressive, for sufficiently large k
  - Finding optimal allocation can become hard even for k=2
  - Furthermore, search occurs with incomplete information

Utility (hyper-)graphs: definition and example

- Each node = one issue under negotiation (i.e. item in a bundle)
- Nodes linked by (hyper-)edges form a cluster
- **Buyer** - cluster potentials:
  u(I1) = $7, u(I2) = $5, u(I3) = $0
  u(I4) = $0, u(I1, I2)= - $5,
  u(I2, I3)=$4, u(I2, I4)=$4



- **Seller** - all items have cost $2.
  u_{BUYER}(I1=0, I2=1, I3=1, I4=1) = $5+$4+$4 = $13
  Gains from Trade = Buyer_utility – Seller_Cost
  Optimal combination?

GT(I1=0, I2=1, I3=1, I4=1)=$13 - 3*$2 = $7

Utility graphs: Use in negotiation

- **Bundles with maximal G.T. ⇔ Pareto-optimal bundles**
  [Somefun, Klos & La Poutre, '04]
- Seller keeps a model of the utility graph of the buyer
- After each offer from the buyer, he updates this model (true graph of the buyer remains hidden)
- He makes a counter-offer by selecting the bundle with the highest *perceived* Gains from Trade

- Seller knows **a maximal utility graph** of possible interdependences (specific to a domain, class of buyers)

Graph partitioning & learning

Selecting the bundle with a maximal GT (w.r.t. to the utility graph learned so far)

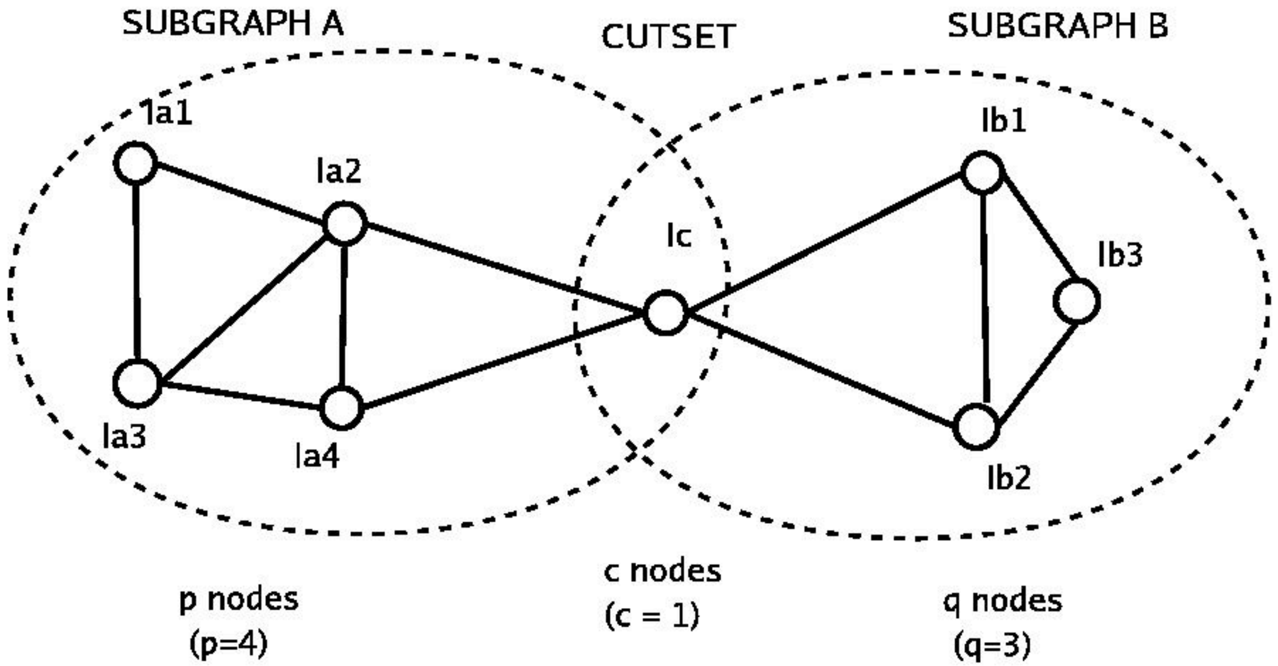
- Exponential problem (e.g. 50 issues: $2^{50} > 10^{15}$ bundles)
- Solved by partitioning into sub-graphs
- Nodes belonging to more than 1 subgraph = cutset nodes
- For all possible instantiations of cutset nodes, compute local sub-bundle combination and merge them

Learning from the opponent's offers

$$u_i(\vec{c}_{i,b}) = u_i(\vec{c}_{i,b}) * (1 + \alpha(i))$$, for the combination induced from buyer's bid

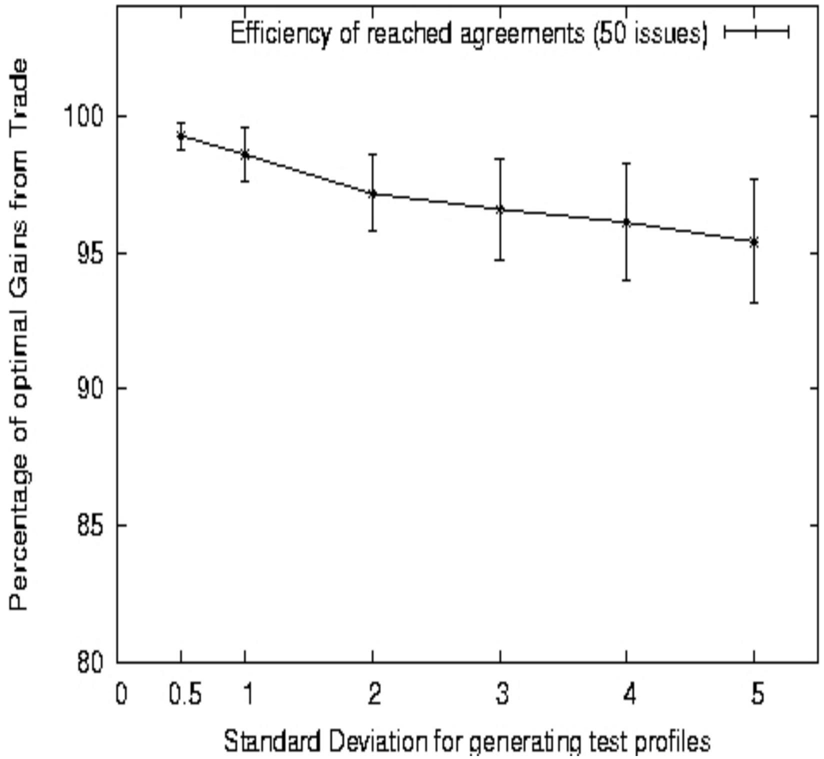
$$u_i(\vec{c}) = u_i(\vec{c}) * (1 - \alpha(i))$$, for all other combinations

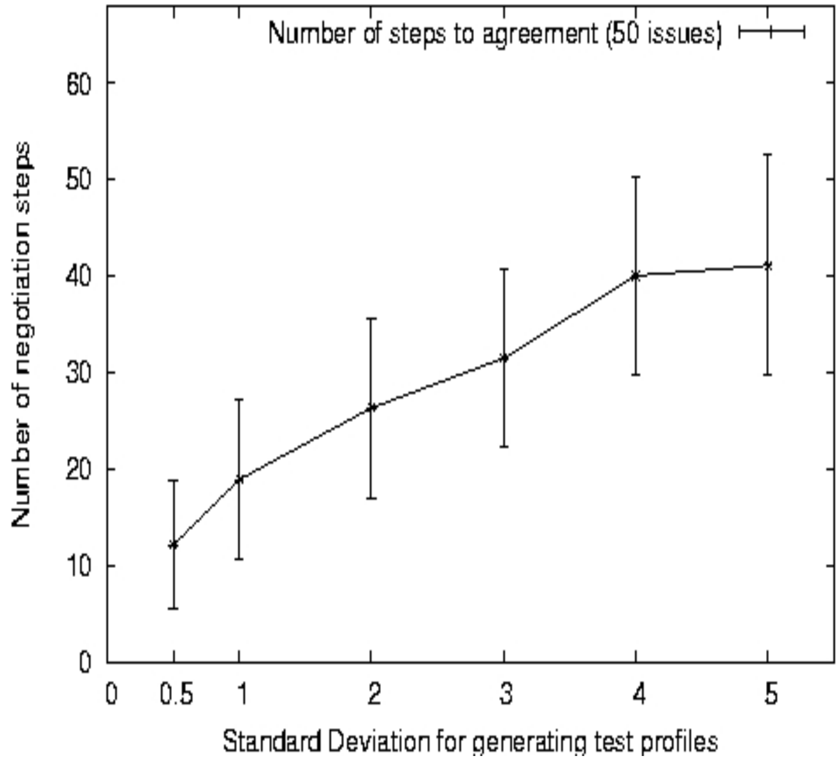
Partitioning a utility graph (example)



- Complexity of exploring all bundles: $2^c * (2^p+2^q)$
- Algorithms for finding balanced partitions exist (minimum k-balanced separator)

Experimental results (50 issues, 75 clusters)





Structure of the initial utility graph

- Preferences of buyers are in some way clustered
- Can we estimate which items can be potentially complementary/substitutable by looking at previous buying patterns?
- **Collaborative filtering** asks the same questions
- Not all relationships hold for all users => only a super-graph is required

Item-based collaborative filtering

- **Item-based similarity:** identifies relationships between items, based on concluded negotiation data
- Several filtering criteria exist

Item-item similarity matrix:

Item pairs	I_1	I_K…	I_{50}
I_1	1	…	0.37
I_K	…	…	…
I_{50}	0.37	…	1

Correlation-based similarity

- For all items i and j: $Sim(i,j) = \frac{\psi_1}{\psi_2}$

$$\psi_1 = N_{i,j}(0,0)Av_i Av_j - N_{i,j}(0,1)Av_i(1-Av_j) - N_{i,j}(1,0)(1-Av_i)Av_j + N_{i,j}(1,1)(1-Av_i)(1-Av_j)$$

$$\psi_2 = \sqrt{\frac{N_i(0)N_i(1)}{N}}\sqrt{\frac{N_j(0)N_j(1)}{N}}$$

- Average buys per item: $Av(i) = \frac{N_i(1)}{N}$

Building the utility super-graph

- Values closer to 1/-1 reflect stronger complementarity/substitutability effects.
- How many dependencies to consider - Trade-off:
  - **Too few:** May affect the outcome at the negotiation stage
  - **Too many:** Introduces too many **spurious dependencies**
- Choice should depend on the **average expected loss** during the negotiation
- Cut-off number of edges – defined as a ratio k of estimated no. of edges to no. of issues

Cut-off point & experiments

- Number of edges considered = k * number of items (vertexes)
- $E_{loss\text{-}GT}(k)=\max \{E_{loss\text{-}GT}(N_{missing}(k)),E_{loss\text{-}GT}(N_{extra}(k))\}$

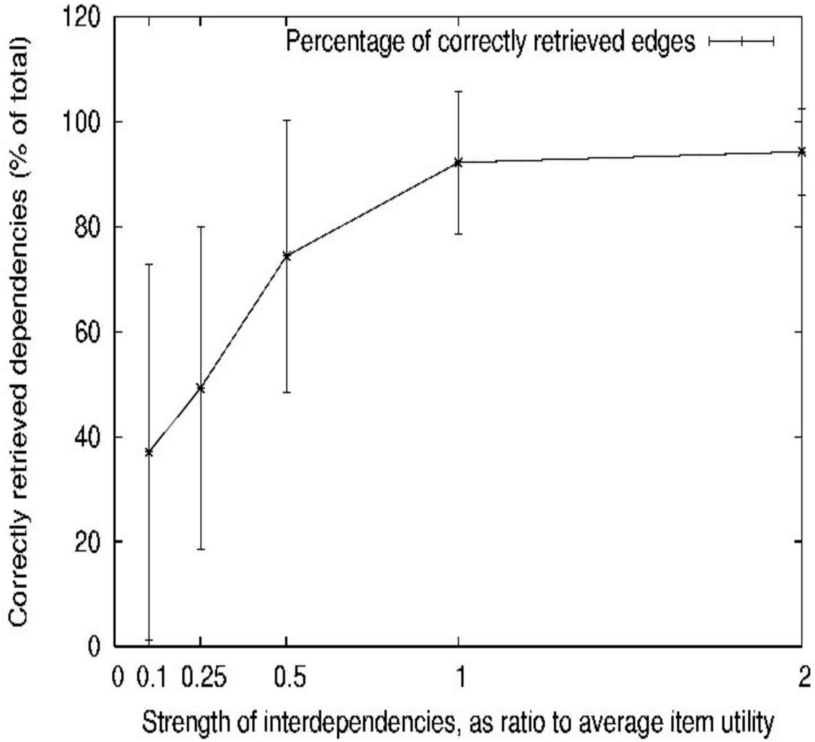
$$K_{opt} = \operatorname{argmin}_K E_{loss\text{-}GT}(k)$$

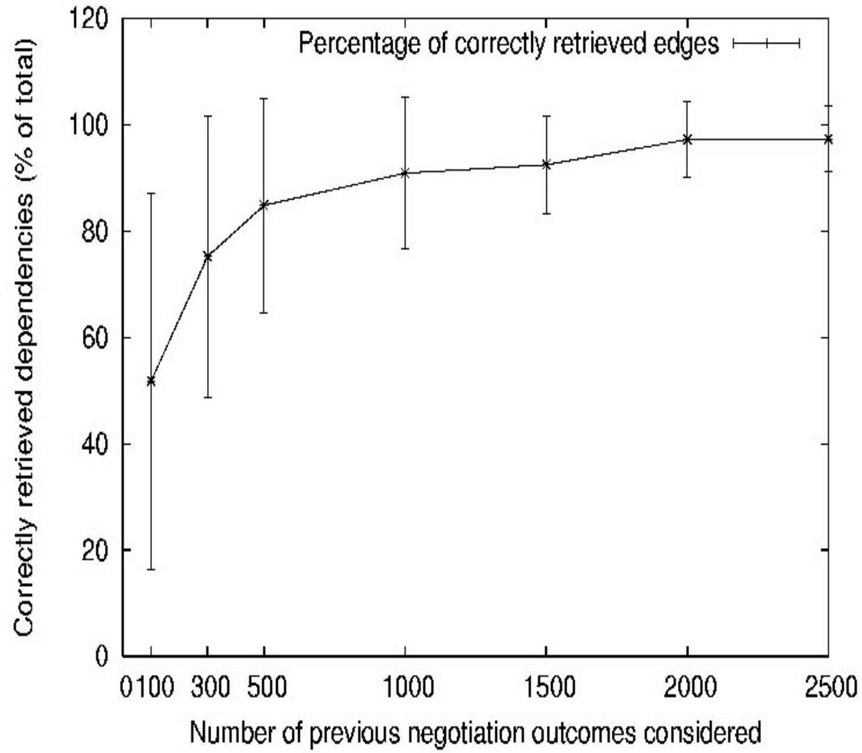
- Intuition: we choose k such as to minimize the expected GT loss ("regret") measure

Experimental set-up:

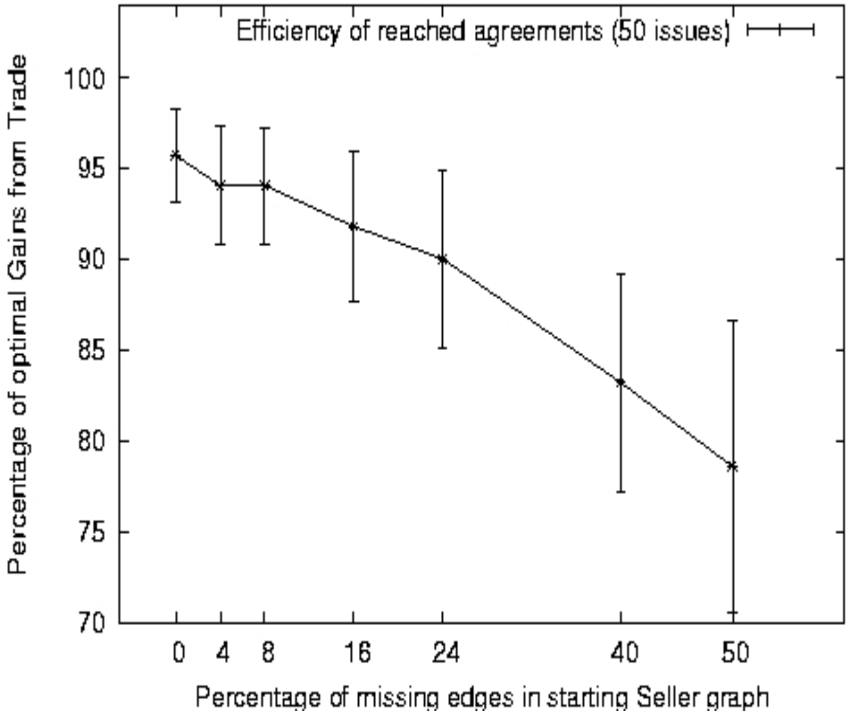
- Graph structure generated at random: for 50 issues 75 binary clusters (50+, 25 -)
- Individual item values drawn from normal i.i.d.-s: N(1, 0-5)).
- Results averaged over 50 tests for each test point

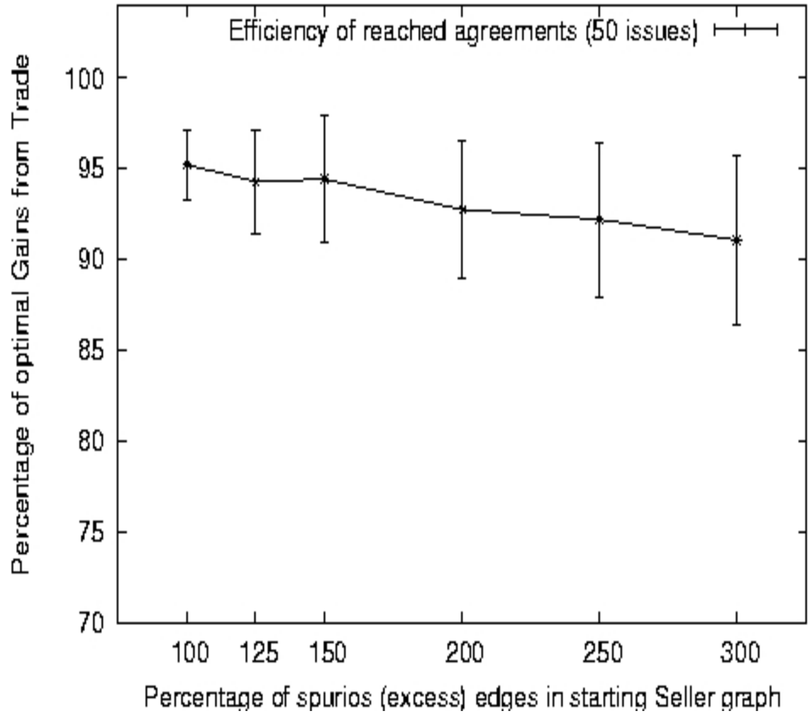
Sensitivity of filtering to negotiation data





Choosing the cut-off size of maximal seller graph





Comparison to other approaches

- Combinatorial auctions: efficient solutions have been proposed for k-additive domains [Conitzer et al. ‘05], but require direct revelation
- Multi-issue negotiation [Klein et al. ‘03] [Lin ‘04]
  - Use simulated annealing & evolutionary
  - No aggregate info. used, all exploration takes place during negotiation
- Preference elicitation
  - 1) Theoretical bound from computational learning theory [Lahaie & Parkes, ’05] (assoc. to polynomial learning)
  - Exact, but computationally expensive (~6500 queries)

Discussion & comparisons

- Preference elicitation (2)
  - [Brazunias & Boutilier, ’05]: based on directed graphs (DAGs)
  - Do not target Pareto efficiency
  - Assumptions on graph structure and value bounds

  Our approach:
  - Negotiation = search for a Pareto-efficient bundle / prices (different aim than exact preference elicitation!)
  - Utilizes the clustering effect between utility functions of typical buyers (filtering part)
  - By combining the two techniques => relatively short negotiations (around 40 steps/50 issues), leading to 90-95% of Pareto-efficiency