Continuous Monitoring of Pareto Frontiers over Partially Ordered Attributes for Many Users

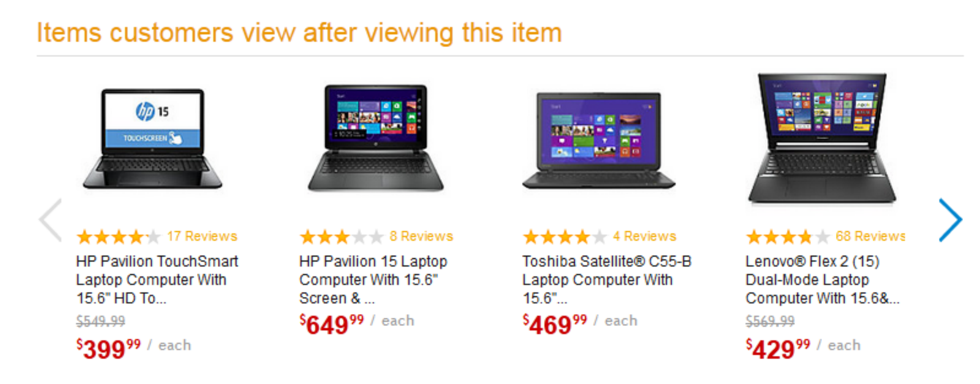
Afroza Sultana and Chengkai Li
Innovative Database and Information Systems Research Lab
The University of Texas at Arlington

EDBT 2018, Vienna, Austria

idir.uta.edu

Motivation

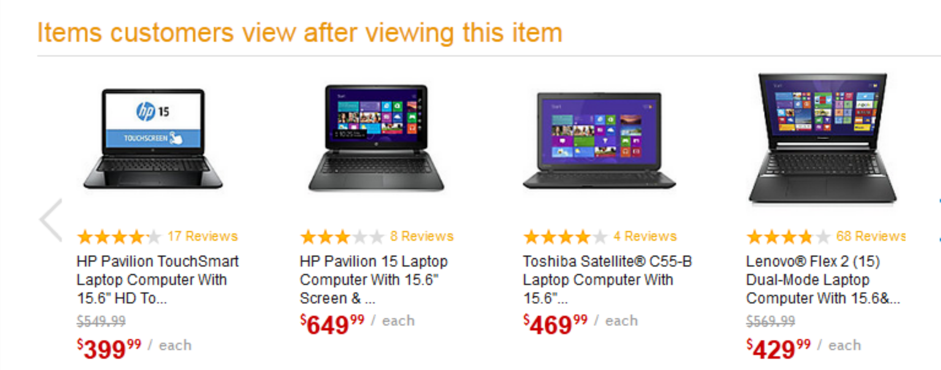
- Recommendation based on users' preferences



Motivation

- Recommendation based on users' preferences
- Preferences with multiple attributes

I prefer 16" to 14" display.

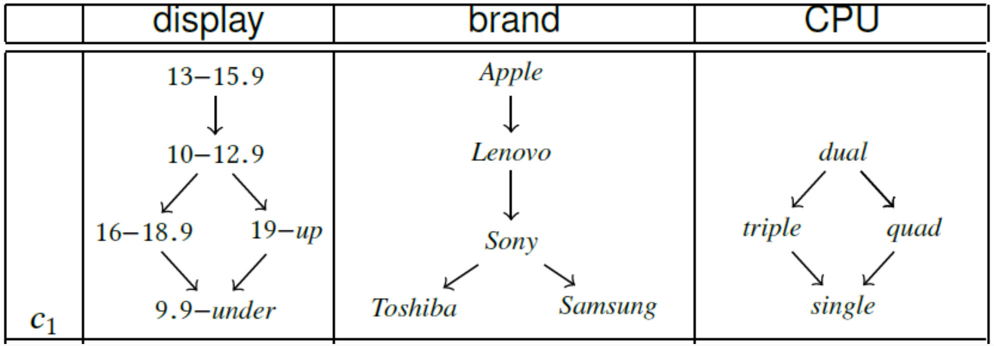


Motivation

- Recommendation based on users' preferences
- Preferences with multiple attributes
- ➢ Goal: objects that "stand out"

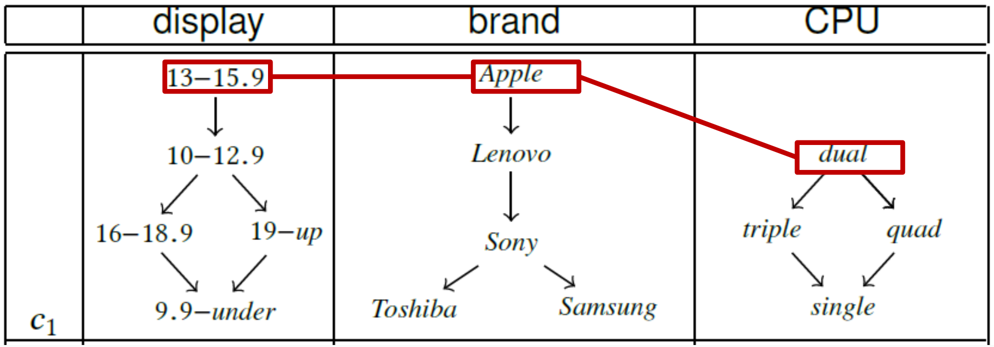
I prefer 16" to 14" display.

An Example



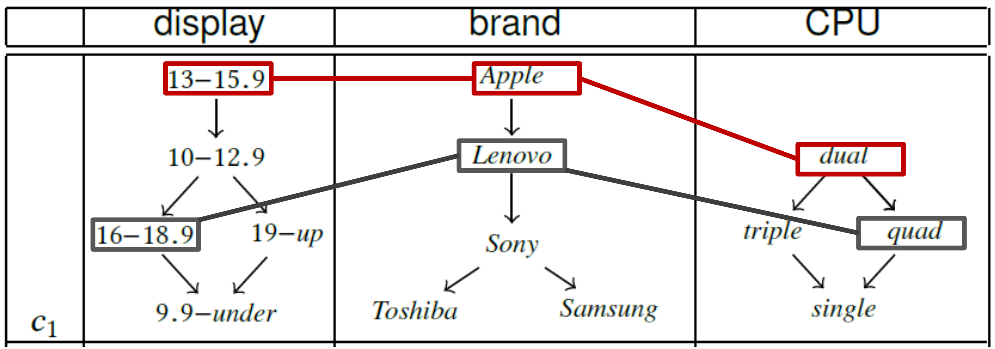
	display	brand	CPU
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
…	…	…	…
o_7	16.5	*Lenovo*	*quad*

An Example



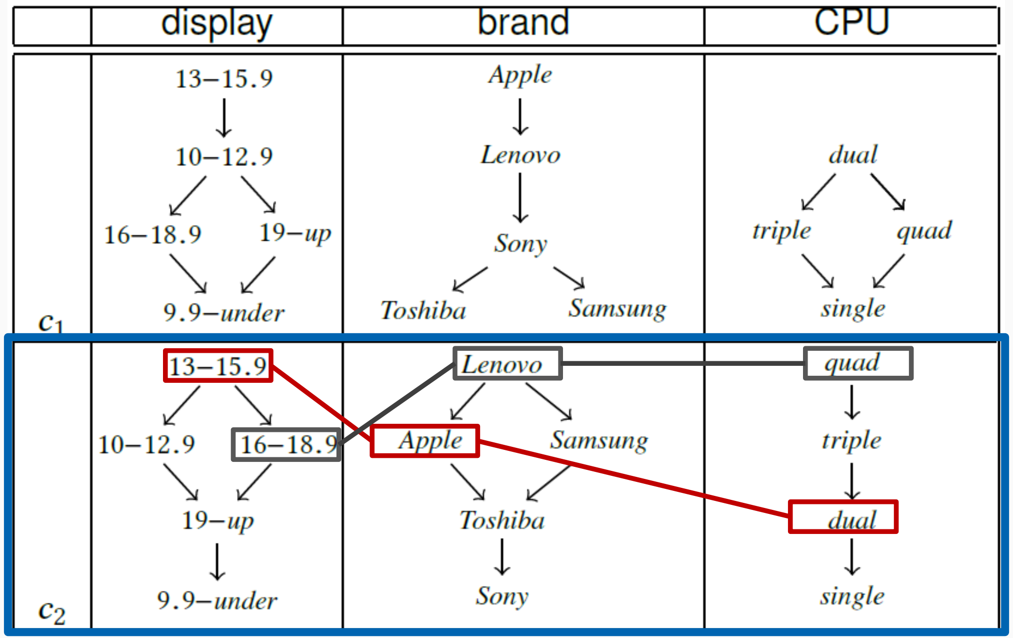
	display	brand	CPU
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
…	…	…	…
o_7	16.5	*Lenovo*	*quad*

An Example



	display	brand	CPU
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
…	…	…	…
o_7	16.5	*Lenovo*	*quad*

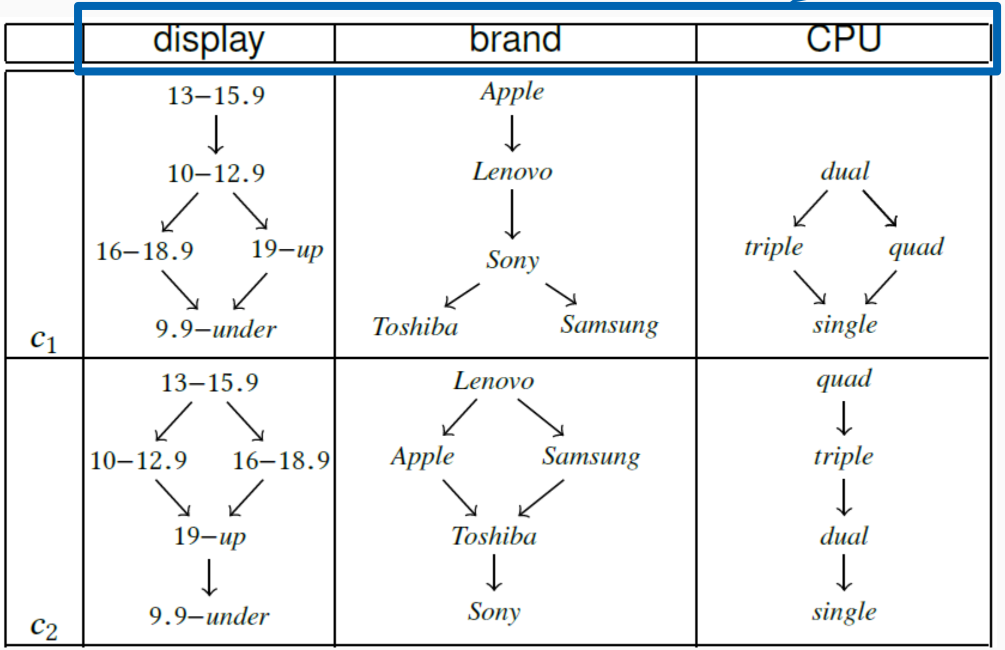
An Example



	display	brand	CPU
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
…	…	…	…
o_7	16.5	*Lenovo*	*quad*

Problem Formulation

Set of attributes $\mathcal{D}$

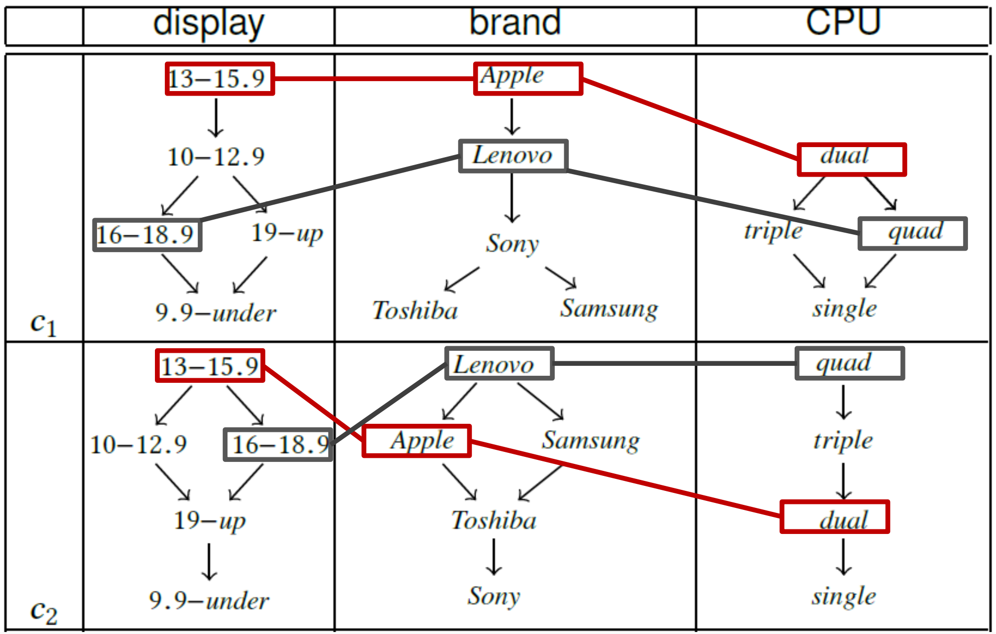


	display	brand	CPU
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
…	…	…	…
o_7	16.5	*Lenovo*	*quad*

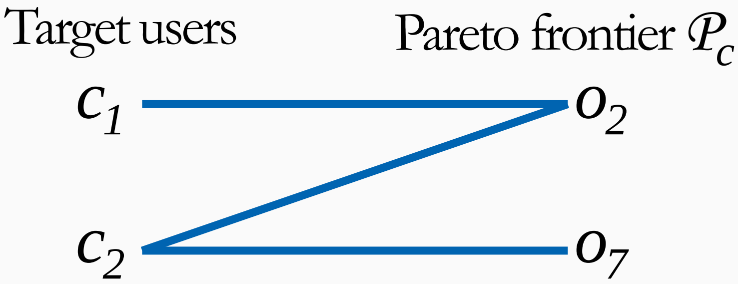
Append-only object table O

Users' preferences

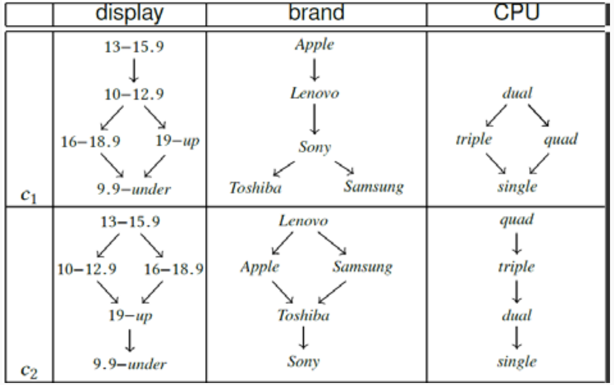
Problem Formulation



	display	brand	CPU
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
…	…	…	…
o_7	16.5	*Lenovo*	*quad*



Problem Formulation; Continuous Object Dissemination

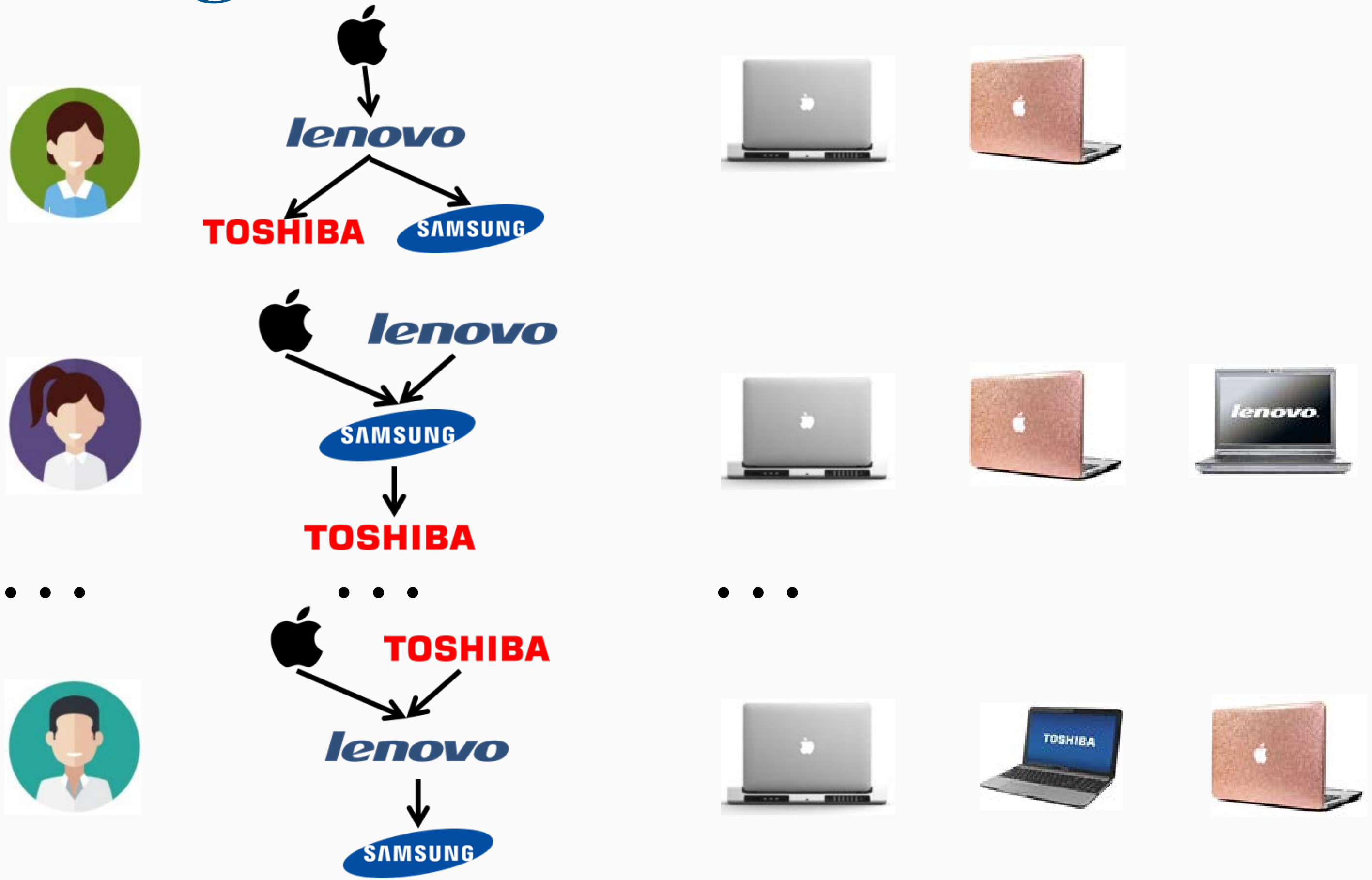


	display	brand	CPU
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
…	…	…	…
o_7	16.5	*Lenovo*	*quad*

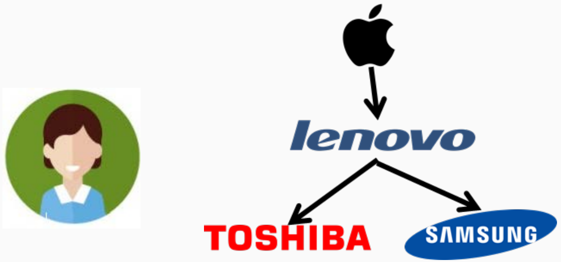
Find target users such that o_7 is in the Pareto frontier.

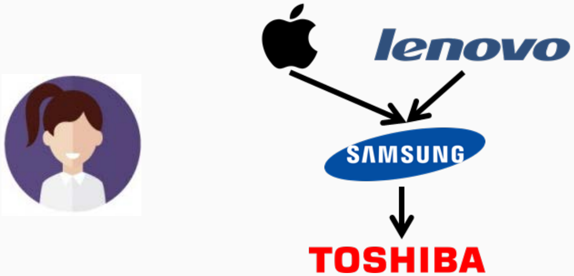
$\boxed{c_2}$

Challenges & Ideas

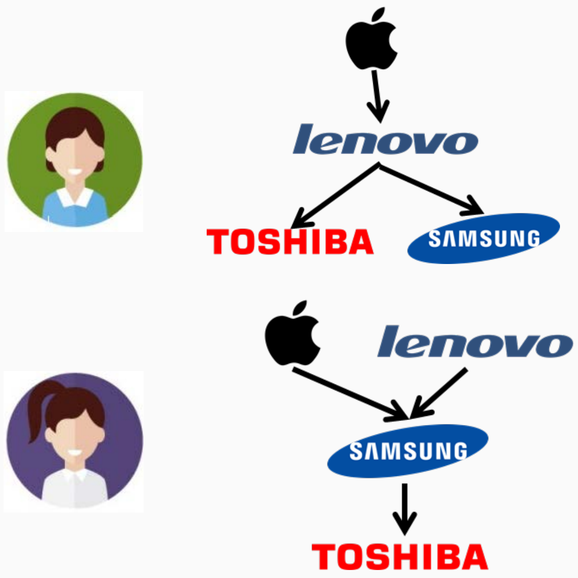


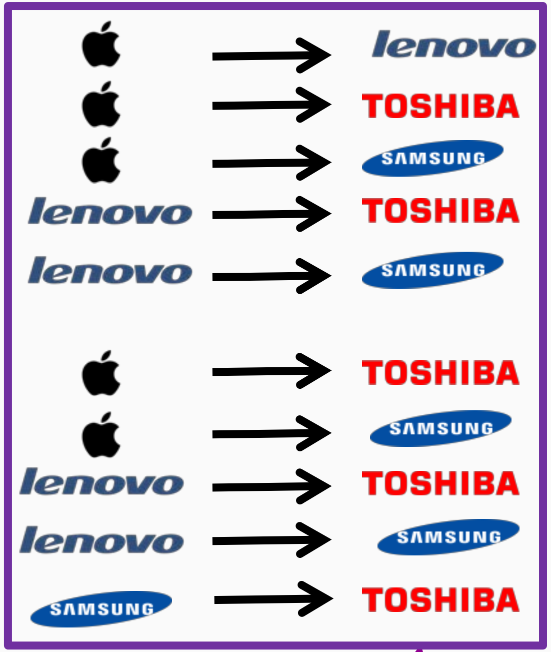
Challenges & Ideas





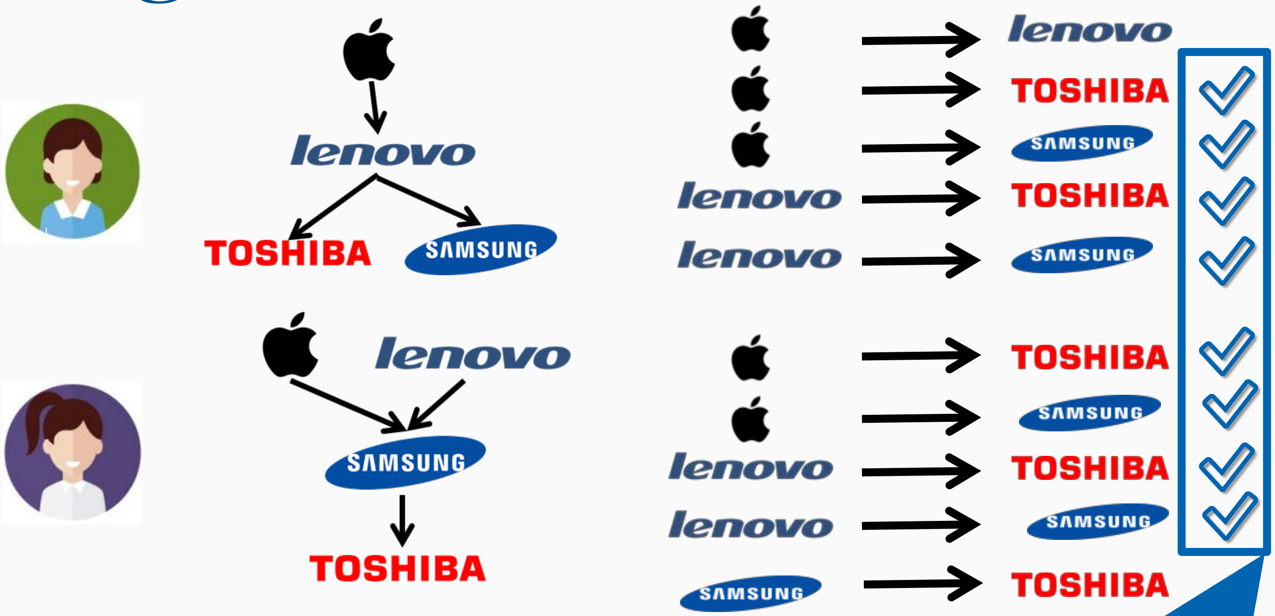
Challenges & Ideas





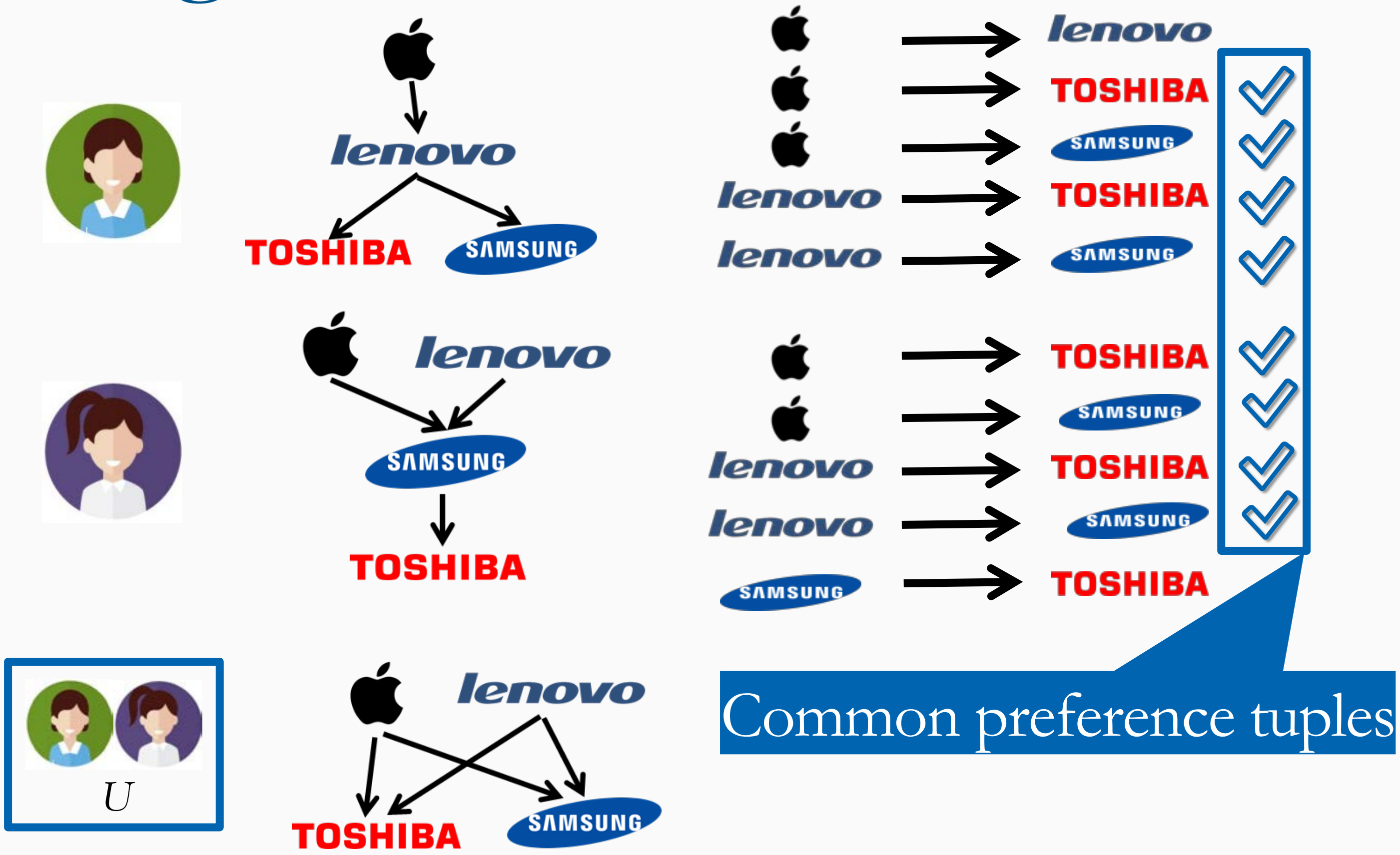
Preference tuples

Challenges & Ideas

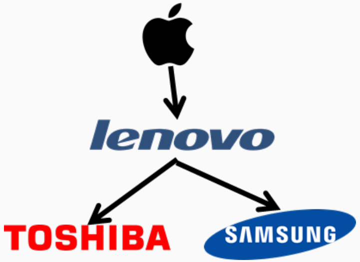


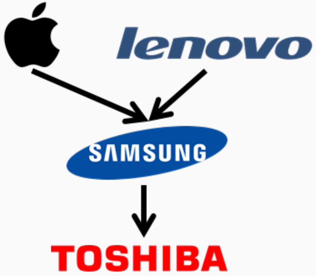
Common preference tuples

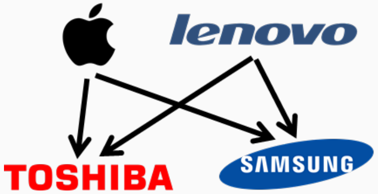
Challenges & Ideas



Challenges & Ideas

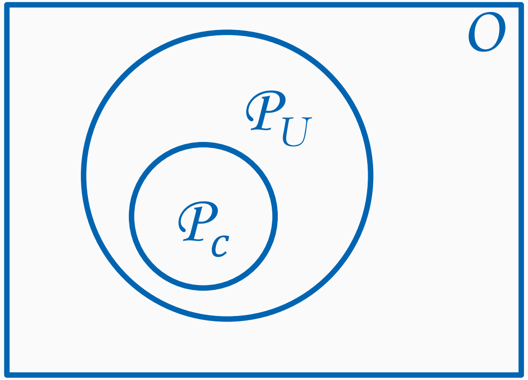






Challenges & Ideas

- Theorem 1: $\mathcal{P}_U \supseteq \mathcal{P}_c$
- Lemma 1: $\mathcal{P}_c$ w.r.t. $O = \mathcal{P}_c$ w.r.t. $\mathcal{P}_U$
- Recall & precision: 100%

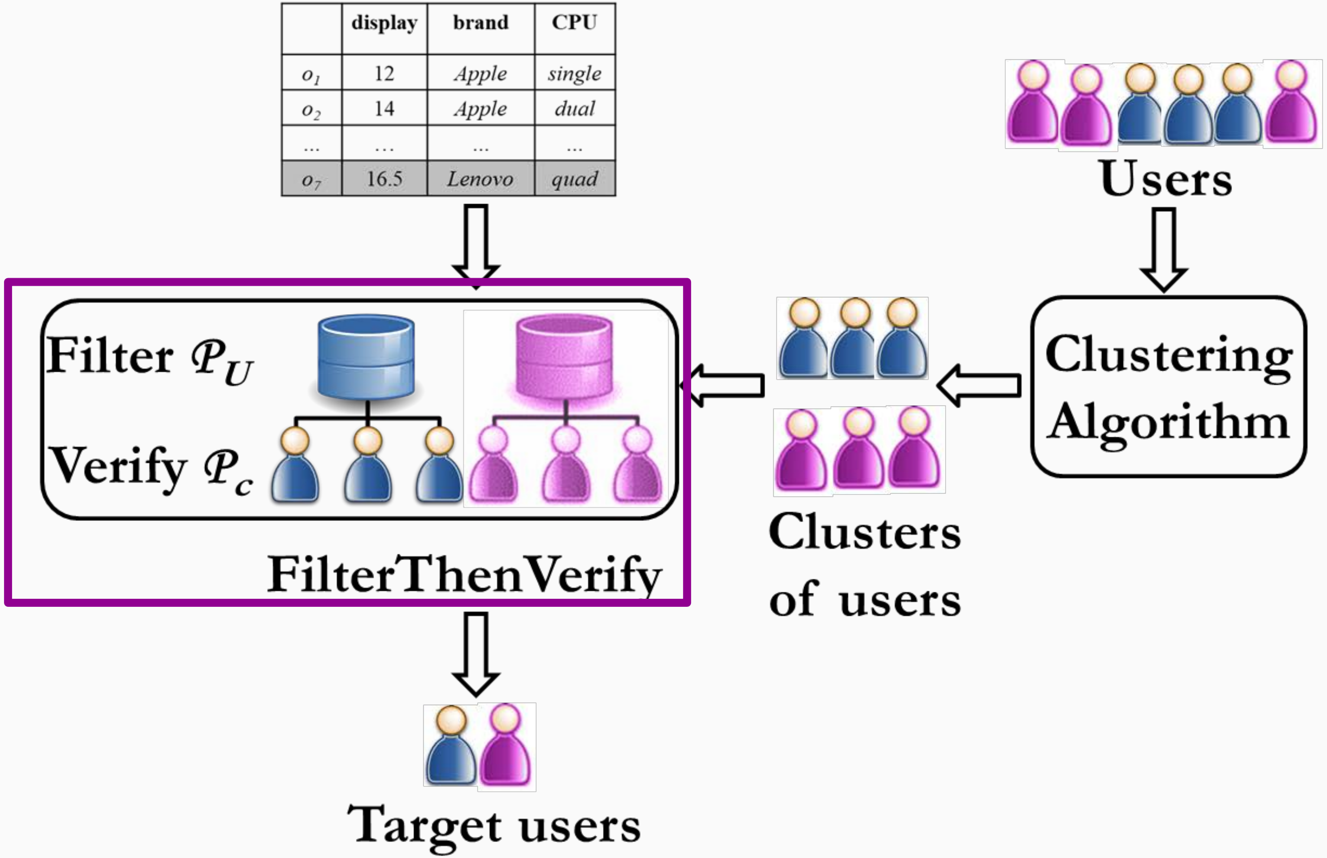


Sharing computation across users

Challenge and Ideas

- ❑ Which users share preferences?
  - ✓ Cluster users based on preferences
- ❑ No prior study on clustering for partial orders
  - ✓ Study clustering partial orders

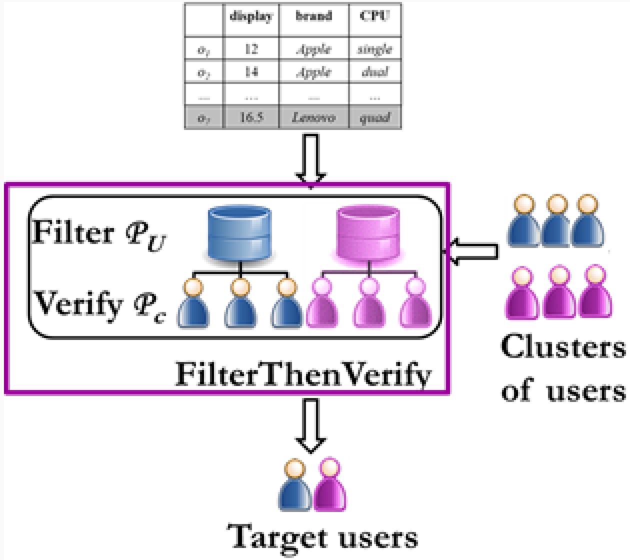
System Architecture



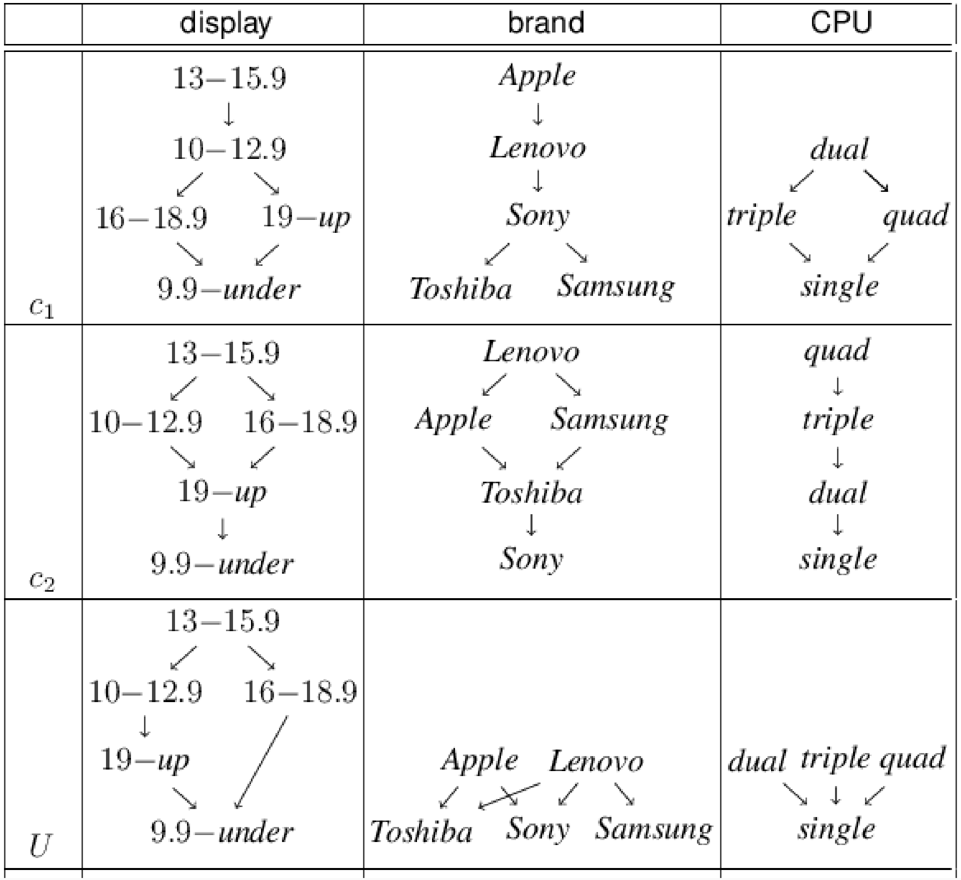
FilterThenVerify

For each cluster in C

- **Filter:** if U approve o in Pareto-optimality, stores o in $\mathcal{P}_U$
  - **Verify:** for each c, determines whether o belongs to $\mathcal{P}_c$

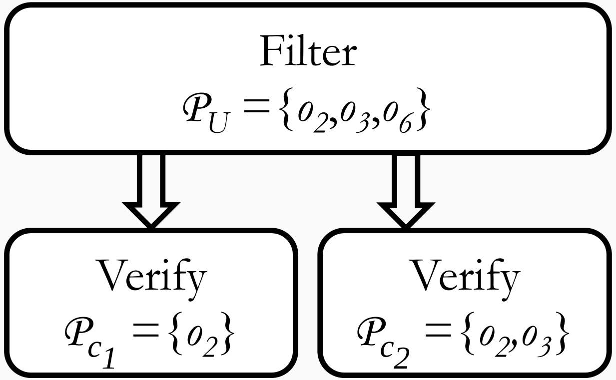


FilterThenVerify



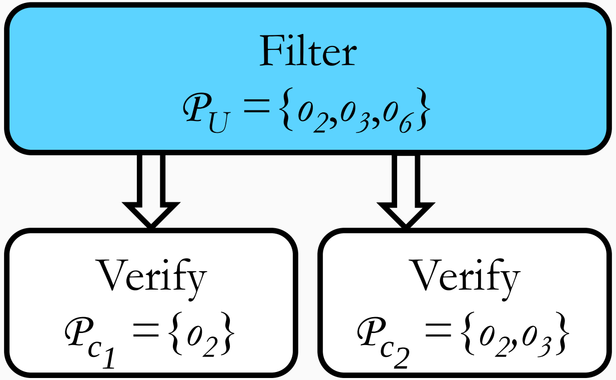
	display	brand	CPU
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
o_3	15	*Samsung*	*dual*
o_4	19	*Toshiba*	*quad*
o_5	9	*Samsung*	*quad*
o_6	9.5	*Lenovo*	*triple*
o_7	16.5	*Lenovo*	*quad*

FilterThenVerify



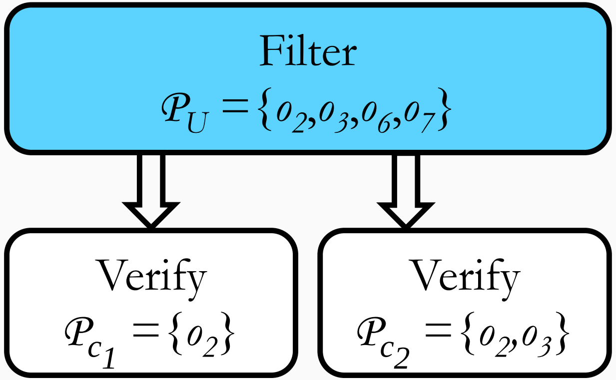
	display	**brand**	**CPU**
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
o_3	15	*Samsung*	*dual*
o_4	19	*Toshiba*	*quad*
o_5	9	*Samsung*	*quad*
o_6	9.5	*Lenovo*	*triple*
o_7	16.5	*Lenovo*	*quad*

FilterThenVerify



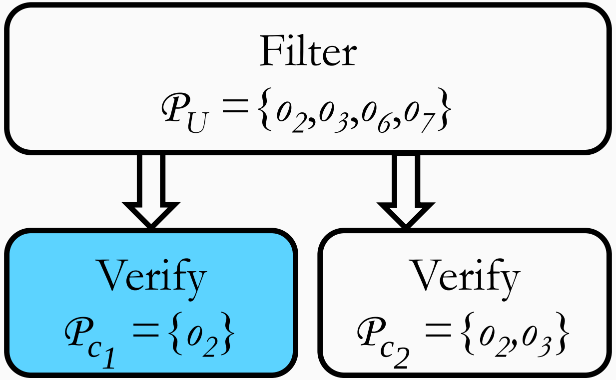
	display	**brand**	**CPU**
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
o_3	15	*Samsung*	*dual*
o_4	19	*Toshiba*	*quad*
o_5	9	*Samsung*	*quad*
o_6	9.5	*Lenovo*	*triple*
o_7	16.5	*Lenovo*	*quad*

FilterThenVerify



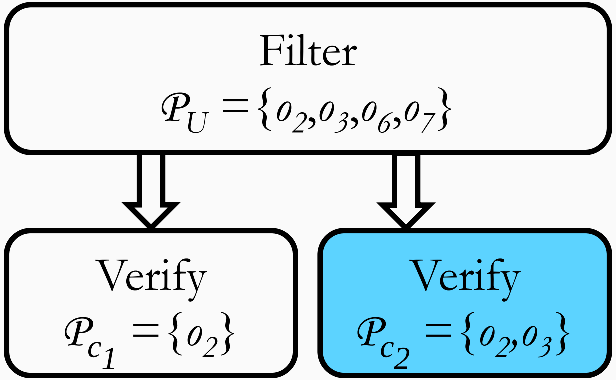
	display	**brand**	**CPU**
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
o_3	15	*Samsung*	*dual*
o_4	19	*Toshiba*	*quad*
o_5	9	*Samsung*	*quad*
o_6	9.5	*Lenovo*	*triple*
o_7	16.5	*Lenovo*	*quad*

FilterThenVerify



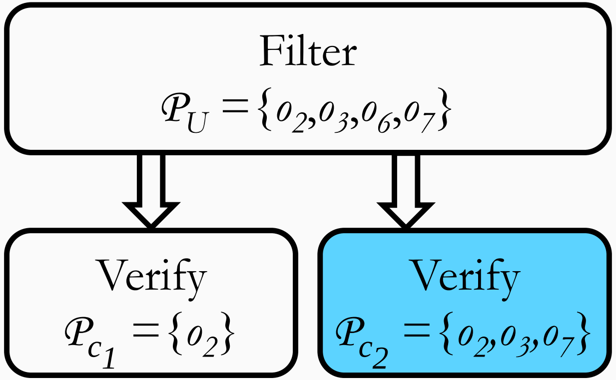
	display	**brand**	**CPU**
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
o_3	15	*Samsung*	*dual*
o_4	19	*Toshiba*	*quad*
o_5	9	*Samsung*	*quad*
o_6	9.5	*Lenovo*	*triple*
o_7	16.5	*Lenovo*	*quad*

FilterThenVerify



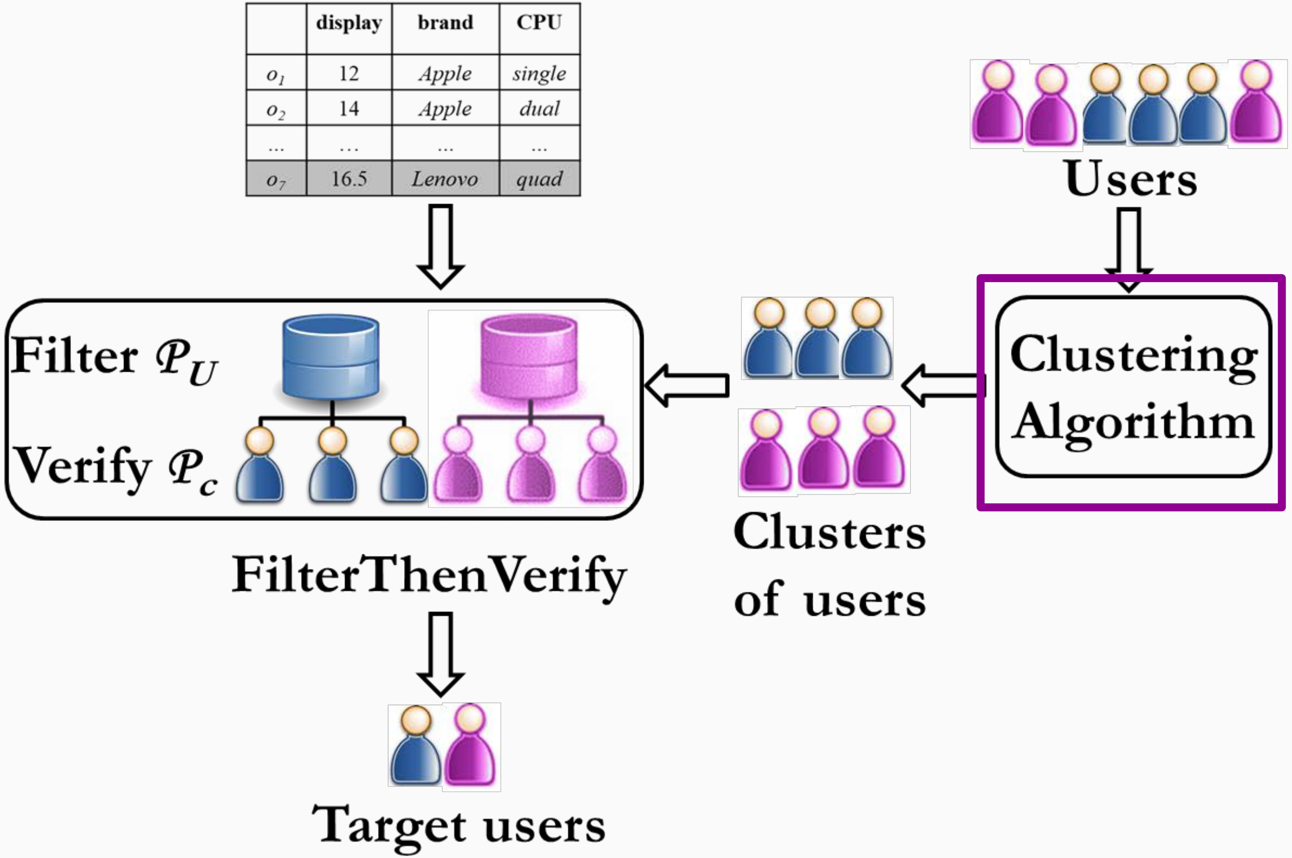
	display	**brand**	**CPU**
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
o_3	15	*Samsung*	*dual*
o_4	19	*Toshiba*	*quad*
o_5	9	*Samsung*	*quad*
o_6	9.5	*Lenovo*	*triple*
o_7	16.5	*Lenovo*	*quad*

FilterThenVerify

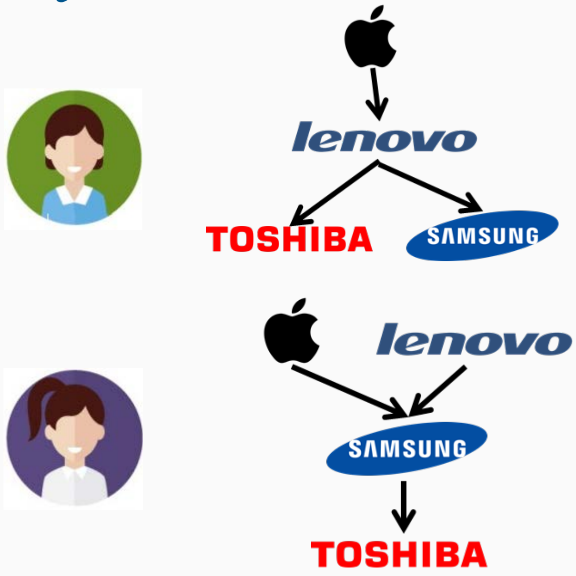


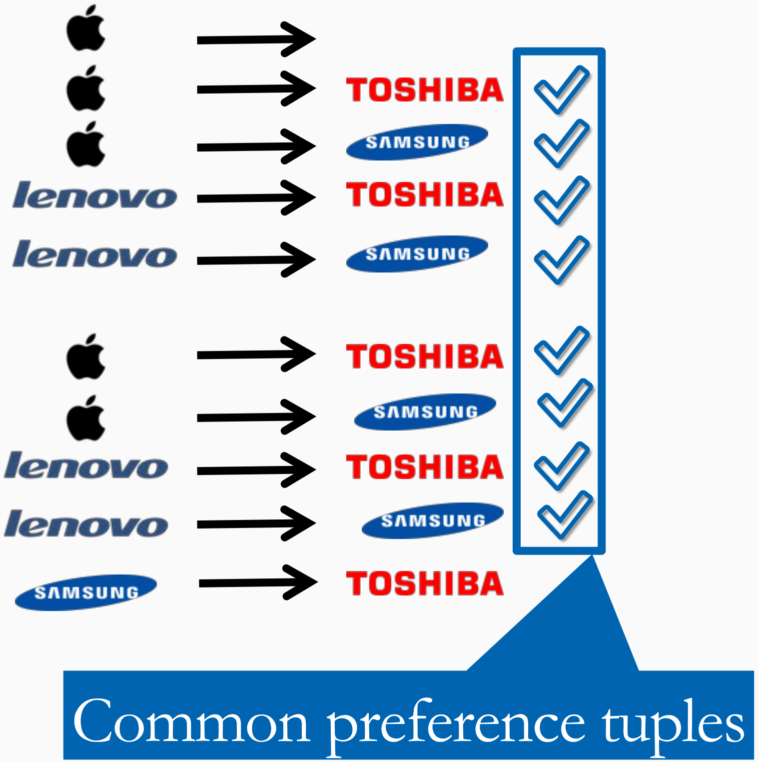
	display	**brand**	**CPU**
o_1	12	*Apple*	*single*
o_2	14	*Apple*	*dual*
o_3	15	*Samsung*	*dual*
o_4	19	*Toshiba*	*quad*
o_5	9	*Samsung*	*quad*
o_6	9.5	*Lenovo*	*triple*
o_7	16.5	*Lenovo*	*quad*

System Architecture



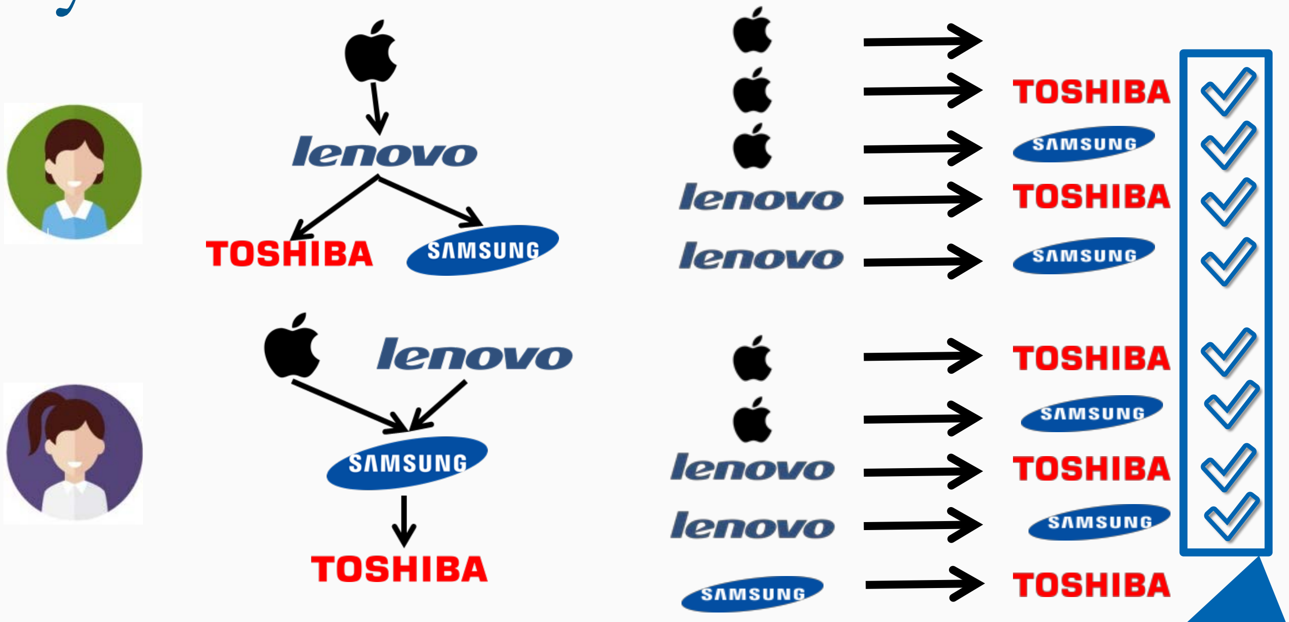
Similarity Function





- Jaccard similarity
  - $\frac{|Common\ preference\ tuples|}{|All\ preference\ tuples|}$

Similarity Function



Common preference tuples

- ❑ Weighted Jaccard similarity
  - ▪Locations of preference tuples

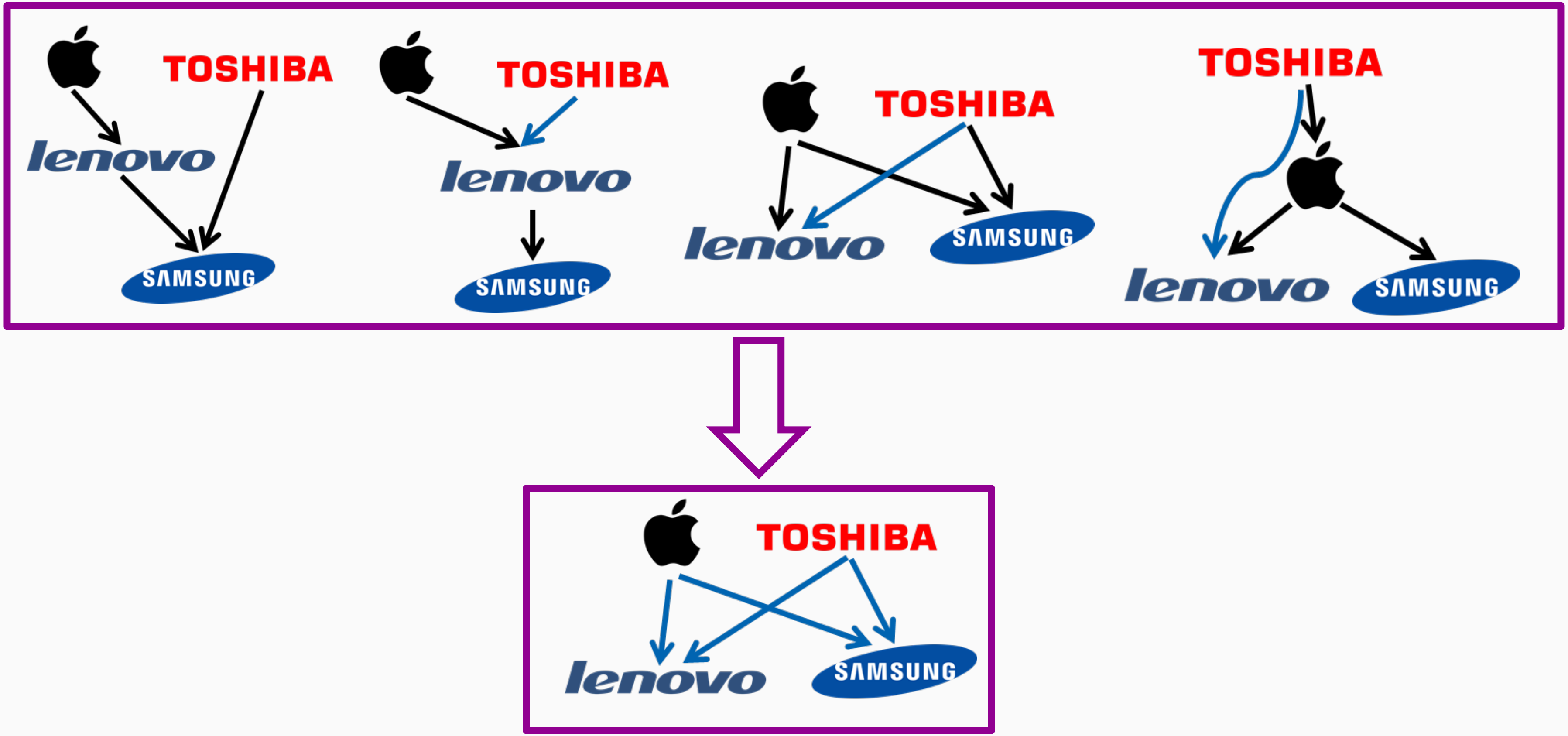
Approx. Common Preference Tuples

- Preferences can be diverse
  - Tiny clusters

Approx. Common Preference Tuples

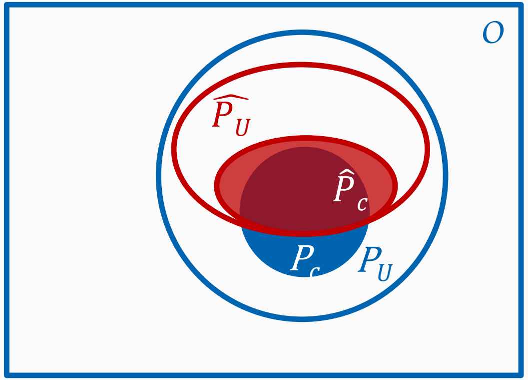
- Preferences can be diverse
  - Tiny clusters
- Relax idea of common preference tuple
  - ✓ Preference polling

GetApproxCommonPreferenceTuples

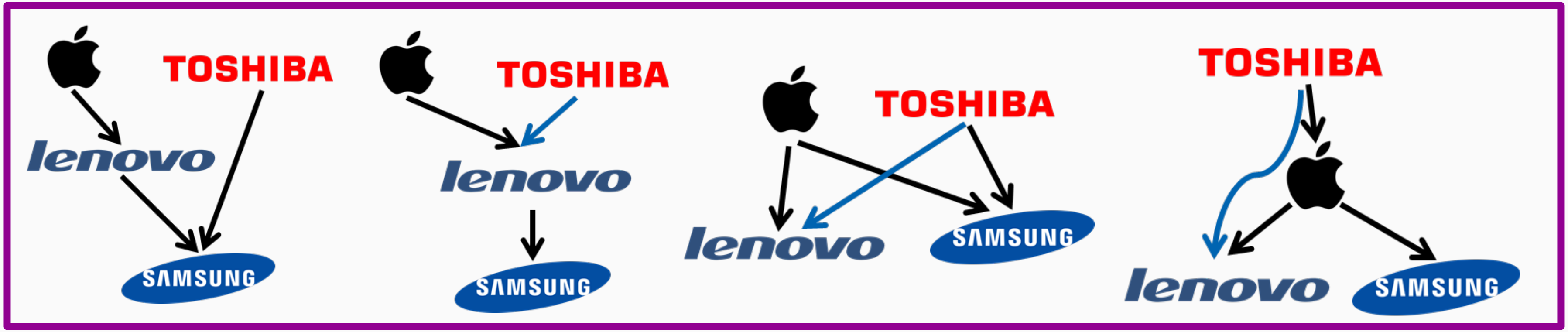


Properties of Approx. Common Preference Tuples

- Pareto frontier w.r.t. approx. common preference tuples: $\widehat{P_U}$
- Pareto frontier w.r.t. user upon approximation: $\widehat{P}_c$
- Lemma 2:
  - Approx. common preference tuples $\supseteq$ Common preference tuples
- Theorem 2
  - $\widehat{P_U} \subseteq P_U$
- Lemma 3
  - $\widehat{P_U} \supseteq \widehat{P}_c$
- Theorem 3
  - $\widehat{P_U} \cap P_c \subseteq \widehat{P}_c$



Similarity Function



- ❑Percentage of preference tuples

Related Works

- ➢Conventional preference query (Kießling VLDB 2002)
  - Pareto frontier w.r.t. individual users, separately
- ✓Our solution---
  - Share computation across multiple users

Related Works

- Mining favorable facets (Wong et al. SIGKDD 2007)
  - Minimum disqualifying condition

	brand	CPU
o_1	*Apple*	*single*
o_2	*Samsung*	*dual*
o_3	*Toshiba*	*quad*

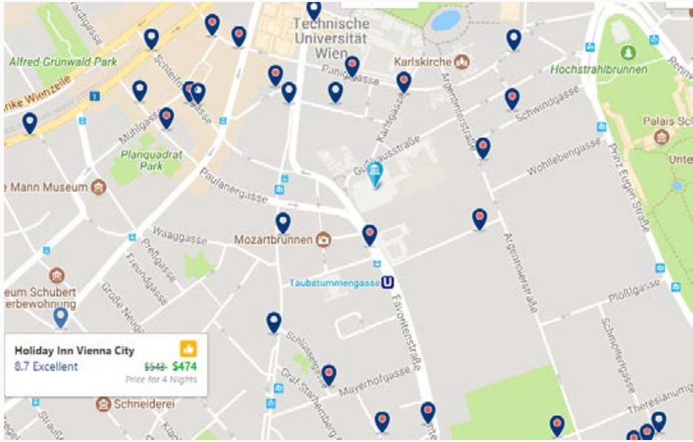
	Minimum set of preferences to disqualify
o_1	((*Samsung*,*Apple*) ∧ (*dual*,*single*)) ∨ ((*Toshiba*,*Apple*) ∧ (*quad*,*single*))
o_2	((*Apple*,*Samsung*) ∧ (*single*,*dual*)) ∨ ((*Toshiba*,*Samsung*) ∧ (*quad*,*dual*))
o_3	((*Apple*,*Toshiba*) ∧ (*single*,*quad*)) ∨ ((*Samsung*,*Toshiba*) ∧ (*dual*,*quad*))

✓ Our solution---

- Compatible with continuously arriving objects

Related Works

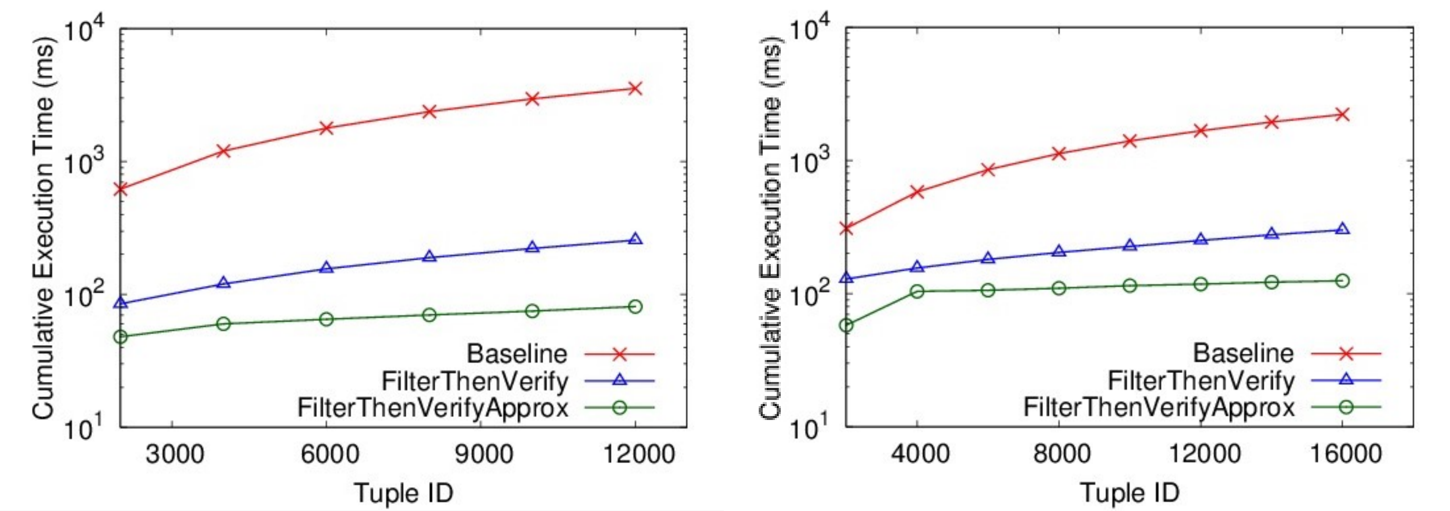
	Attribute	Order
Reverse skyline query (Dellis et al. VLDB 2007)	Numerical: price, distance	Total
Our solution	Categorical/numerical: brand, hotel/suite	Partial



Experiment by Simulation

- Movie Dataset
  - 12,749 movies: joined Netflix dataset with data from IMDB
  - 1000 users
  - 4 attributes: *actor*, *director*, *genre*, *writer*
- Publication Dataset
  - 17,598 publications: ACM Digital Library
  - 1000 users
  - 4 attributes: *affiliation*, *author*, *conference*, and *keyword*

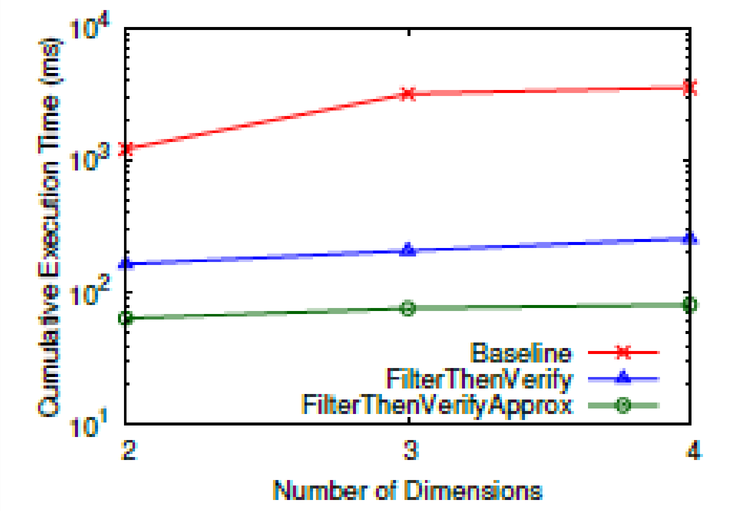
Performance of FilterThenVerify/FilterThenVerifyApprox



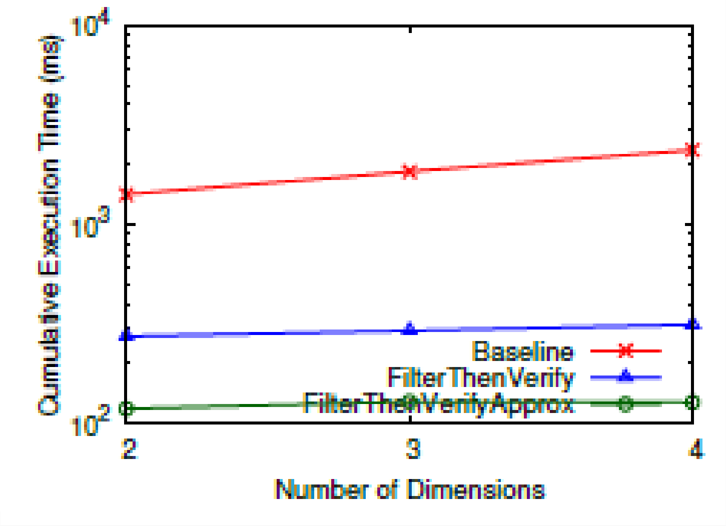
Movie Dataset **Publication Dataset**

- Baseline < FilterThenVerify/FilterThenVerifyApprox
  - Fewer comparisons due to filtering
- FilterThenVerify < FilterThenVerifyApprox
  - Approx. allows more sharing

Performance of FilterThenVerify/FilterThenVerifyApprox



Movie Dataset



Publication Dataset

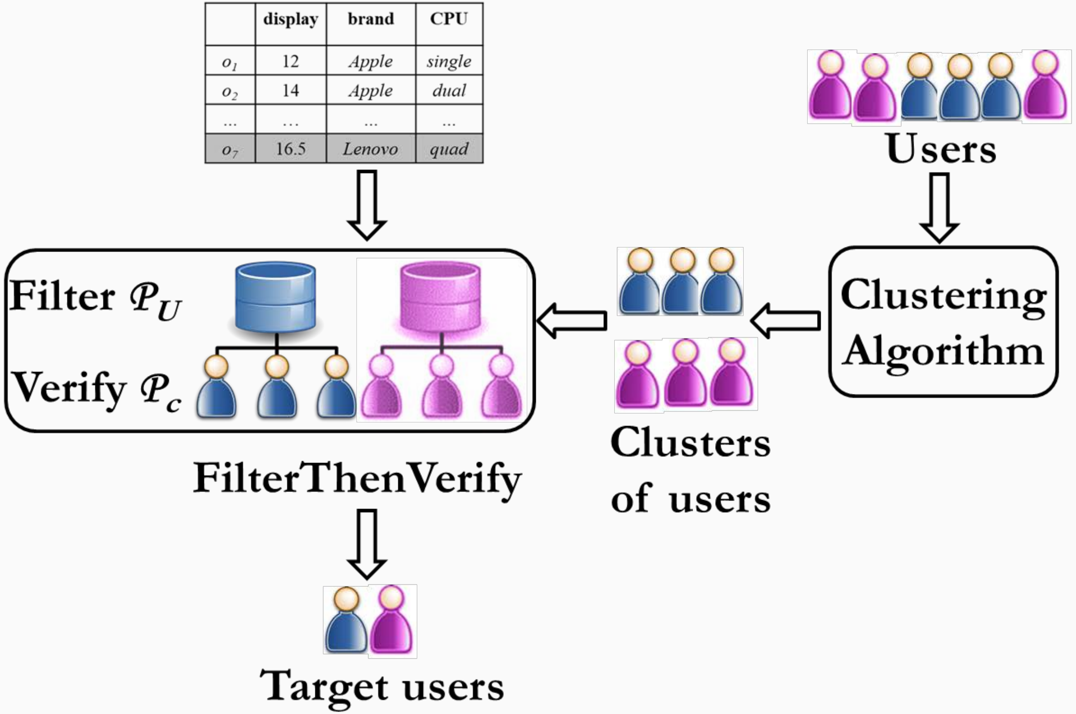
- Execution time increases with *d*
  - High *d*=>large Pareto frontiers=>more comparisons

Efficacy of FilterThenVerifyApprox

Dataset	h = 0.70			h = 0.55		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Movie	100	95.43	97.67	99.99	90.46	94.99
Publication	100	96.59	98.27	100	95.13	97.51

- Recall decreases with h
  - Small h=>large clusters=>high false negatives
- Stable precision
  - Few false negatives=>fewer false positives

Conclusion



- ✓ Efficient algorithm to find target users
- ✓ Novel problem of clustering partial orders

THANK YOU!