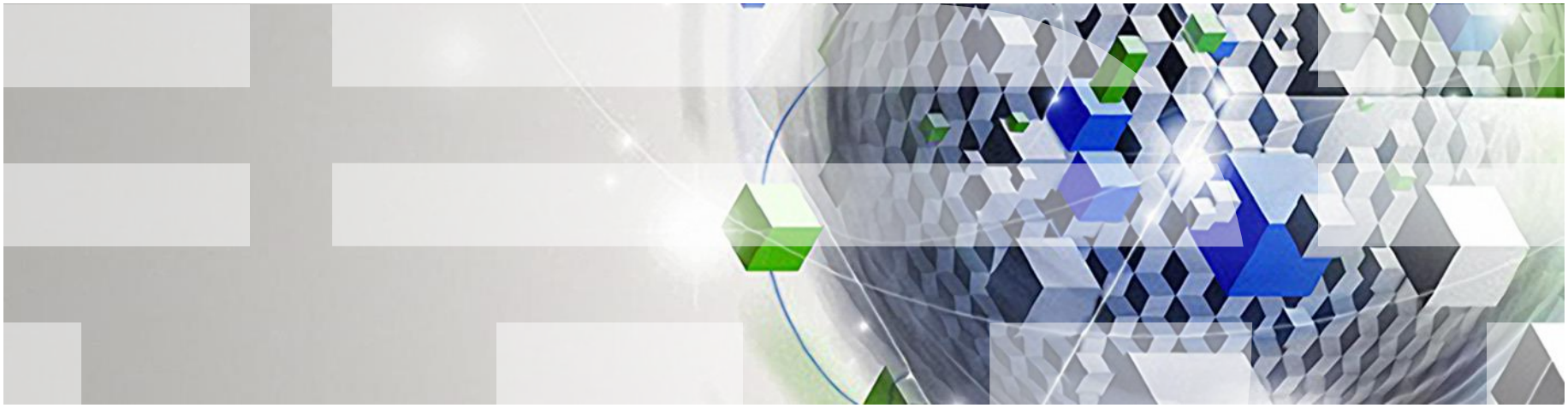
Effective and Complete Discovery of Order Dependencies via Set-based Axioms

Jarek Szlichta, Parke Godfrey, Lukasz Golab,
Mehdi Kargar, Divesh Srivastava

University of Ontario Institute of Technology, Canada; York University, Canada
University of Waterloo, Canada; University of Windsor, Canada
AT&T Labs-Research, US



Introduction

- Integrity constraints are commonly used to characterize **data quality** and **optimize business queries**
- Unlike functional dependencies (FDs), order dependencies (ODs) capture relationships among attributes with naturally **ordered domains** such as timestamps, numbers and strings
  - E.g., An employee that pays higher taxes has a higher salary than another employee (`salary` orders `tax`)

Background: Order Dependencies

- The **order dependency (OD)** **X** ↦ **Y** means that **Y**'s list attribute values are monotonically non-decreasing wrt **X**'s list attribute values (**if the dataset is sorted by X, it is also sorted by Y**)
  - E.g., `sal` ↦ `tax` and `sal` ↦ `grp, subg` hold in employee salary table
  - **Order Compatible** **X** ∼ **Y** if **XY** ↔ **YX**

#	ID	yr	posit	bin	sal	perc	tax	grp	subg
t1	10	16	secr	1	5K	20%	1K	A	III
t2	11	16	mngr	2	8K	25%	2K	C	II
t3	12	16	direct	3	10K	30%	3K	D	I
t4	10	15	secr	1	4.5K	20%	0.9K	A	III
t5	11	15	mngr	2	6K	25%	1.5K	C	I
t6	12	15	direct	3	8K	25%	2K	C	II

OD Violations

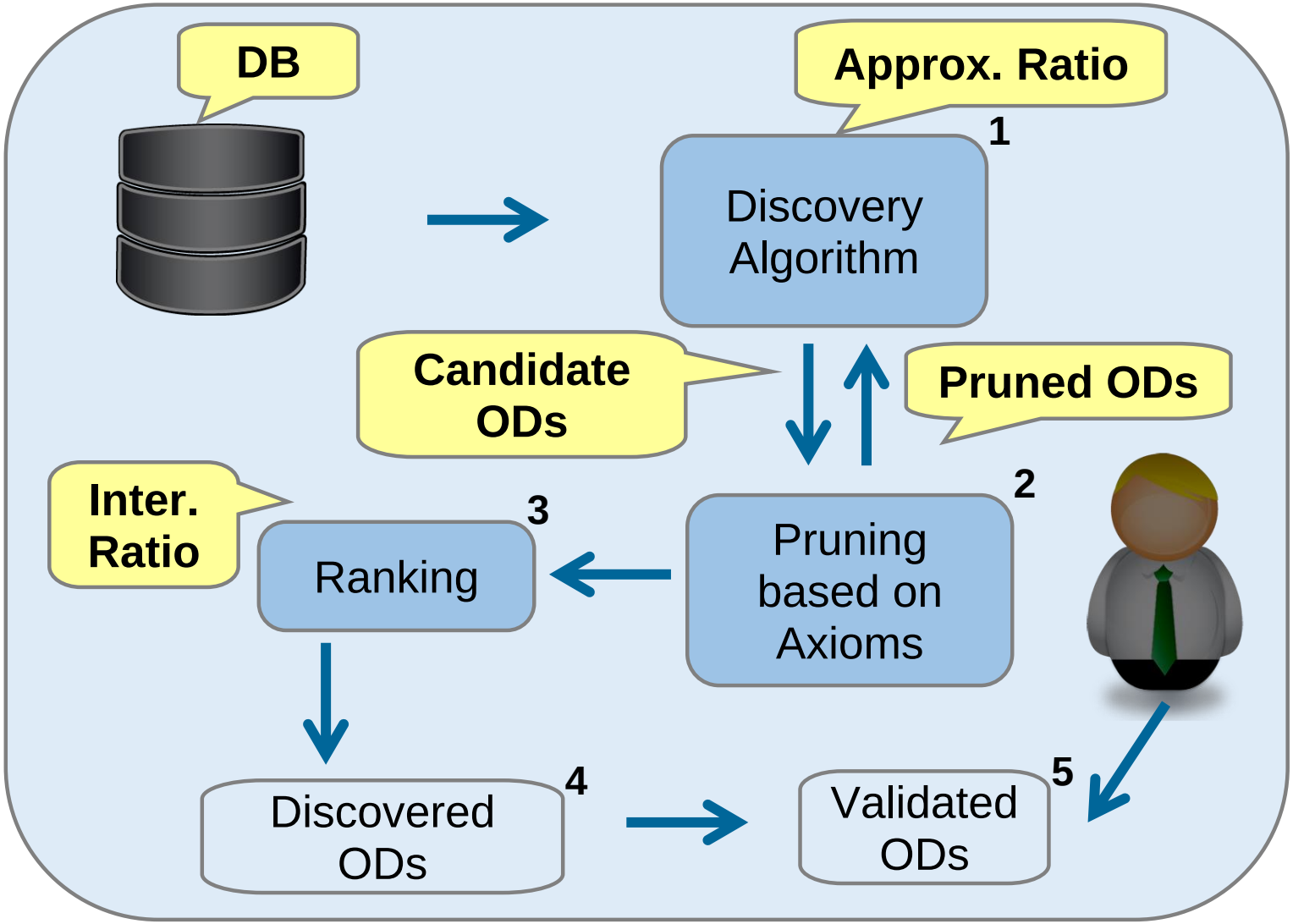
- However, the OD `sal` ↦ `subg, grp` does NOT hold in the following employee salary table
  - **illustrating that the order of attributes in ODs matters!**

#	ID	yr	posit	bin	sal	perc	tax	grp	subg
t1	10	16	secr	1	5K	20%	1K	A	III
t2	11	16	mngr	2	8K	25%	2K	C	II
t3	12	16	direct	3	10K	30%	3K	D	I
t4	10	15	secr	1	4.5K	20%	0.9K	A	III
t5	11	15	mngr	2	6K	25%	1.5K	C	II
t6	12	15	direct	3	8K	25%	2K	C	I

Contributions

- Our aim is to provide a deep understanding of OD discovery from real-datasets
  - We address the limitations of prior work, which has factorial complexity in the number of attributes
  - An insight we present is that list-based ODs can be expressed using sets of attributes via a polynomial mapping to set-based canonical-form
  - The mapping allows us to design an efficient OD discovery algorithm that has exponential worst-case complexity in the number of attributes
    - This complexity is similar to previous FD discovery algorithms such as TANE

Framework Overview



Canonical Form: Constant

- Attribute `A` is a **constant** within each equivalence class wrt set of attributes *X* called context if there is no **splits**, where values of `A` are not equal, denoted as X: [] ↦ `A`
  - E.g, `bin` **is a constant** in the context of `posit`, i.e., **{posit}: [] ↦ bin**
  - But salary **is NOT a constant** in the context of posit, i.e., **{posit}: [] ↦ sal** does not hold

#	ID	yr	posit	bin	sal	perc	tax	grp	subg
t1	10	16	secr	1	5K	20%	1K	A	III
t2	11	16	mngr	2	8K	25%	2K	C	II
t3	12	16	direct	3	10K	30%	3K	D	I
t4	10	15	secr	1	4.5K	20%	0.9K	A	III
t5	11	15	mngr	2	6K	25%	1.5K	C	I
t6	12	15	direct	3	8K	25%	2K	C	II

Canonical Form: Order-Compatibility

- Two marked attributes `A` and `B` are **order-compatible** if there are no **swaps** where `A` grows and `B` goes down within each equivalence class *X*, denoted as `X: A ~ B`
  - E.g., there is no swap between `bin` and `sal` in the context of `year`, i.e., **{year}: bin ~ sal**
  - But there is a swap between `bin` and `subg` in the context of `year`, i.e., **{year}: bin ~ subg** does not hold

#	ID	yr	posit	bin	sal	perc	tax	grp	subg
t1	10	16	secr	1	5K	20%	1K	A	III
t2	11	16	mngr	2	8K	25%	2K	C	II
t3	12	16	direct	3	10K	30%	3K	D	I
t4	10	15	secr	1	4.5K	20%	0.9K	A	III
t5	11	15	mngr	2	6K	25%	1.5K	C	I
t6	12	15	direct	3	8K	25%	2K	C	II

Canonical Form: Mapping

- Polynomial Mapping of list-based ODs to equivalent set-based ODs, given **X** ↦ **Y**
  - In the context of *X* all attributes in *Y* are constants
  - In the context of all prefixes over **XY** the following attributes are order compatible
- E.g., an OD [AB] ↦ [CD] can be mapped into:
  - {A,B}: [] ↦ C, {A,B}: [] ↦ D
  - {}: A ~ C, {A}: B ~ C, {C}: A ~ D, {A, C}: B ~ D

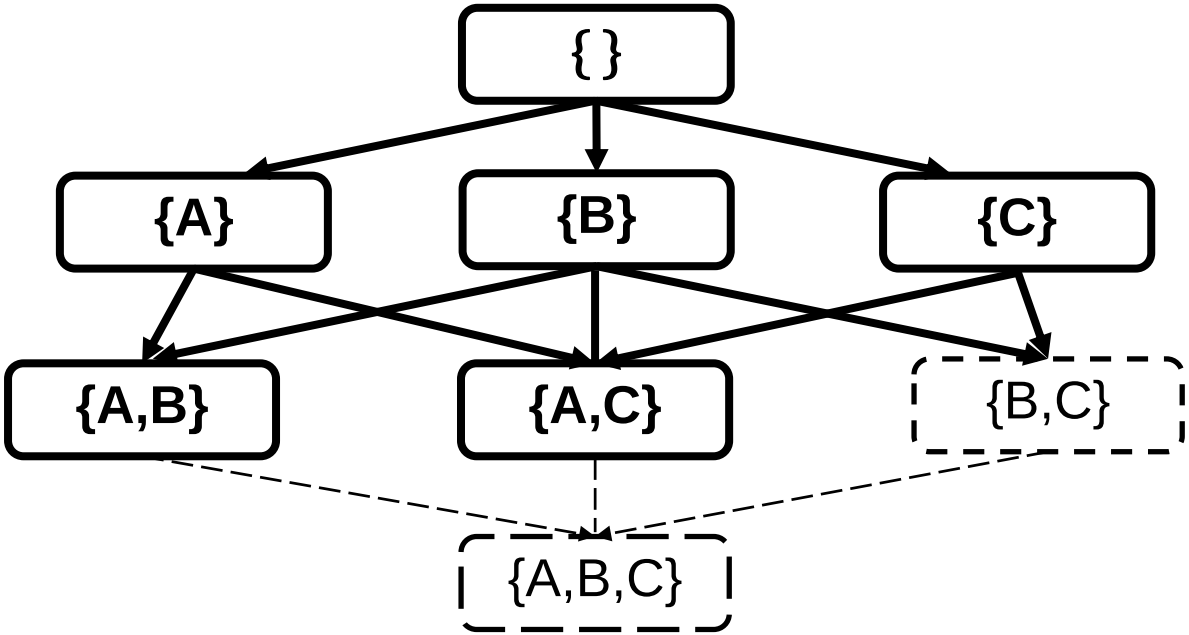
Set-based Axiomatization for Canonical ODs

1. Reflexivity
$$\mathcal{X}\text{: } [] \mapsto A, \forall A \in \mathcal{X}$$
2. Identity
$$\mathcal{X}\text{: } A \sim A$$
3. Commutativity
$$\frac{\mathcal{X}\text{: } A \sim B}{\mathcal{X}\text{: } B \sim A}$$
4. Strengthen
$$\frac{\mathcal{X}\text{: } [] \mapsto A \quad \mathcal{X}A\text{: } [] \mapsto B}{\mathcal{X}\text{: } [] \mapsto B}$$
5. Propagate
$$\frac{\mathcal{X}\text{: } [] \mapsto A}{\mathcal{X}\text{: } A \sim B}$$
6. Augmentation-I
$$\frac{\mathcal{X}\text{: } [] \mapsto A}{\mathcal{Z}\mathcal{X}\text{: } [] \mapsto A}$$
7. Augmentation-II
$$\frac{\mathcal{X}\text{: } A \sim B}{\mathcal{Z}\mathcal{X}\text{: } A \sim B}$$
8. Chain
$$\frac{\mathcal{X}\text{: } A \sim B_1 \quad \forall_{i\in[1,n-1]}, \mathcal{X}\text{: } B_i \sim B_{i+1} \quad \mathcal{X}\text{: } B_n \sim C \quad \forall_{i\in[1,n]}, \mathcal{X}B_i : A \sim C}{\mathcal{X}\text{: } A \sim C}$$

- We show the **axiomatization is sound and complete** for set-based ODs

Lattice

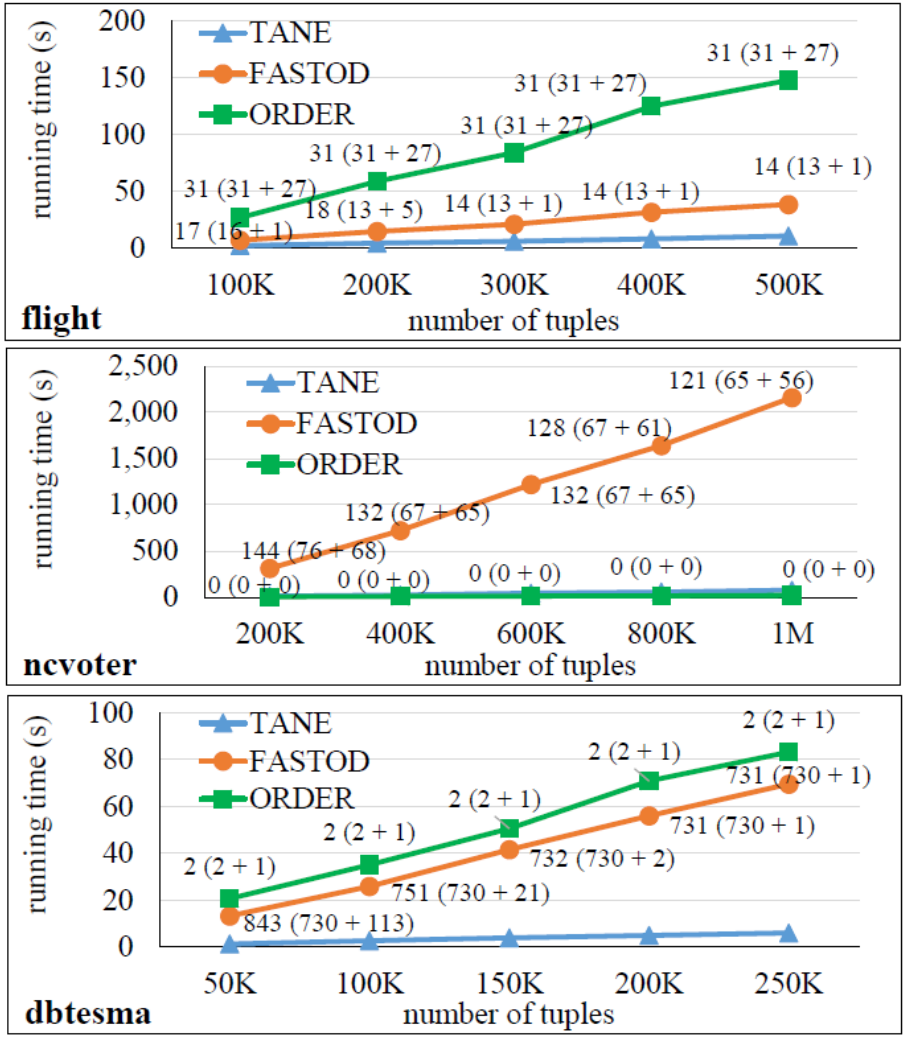
- FASTOD traverses the lattice until all complete and minimal ODs are found
- We use the axioms to prune the search space



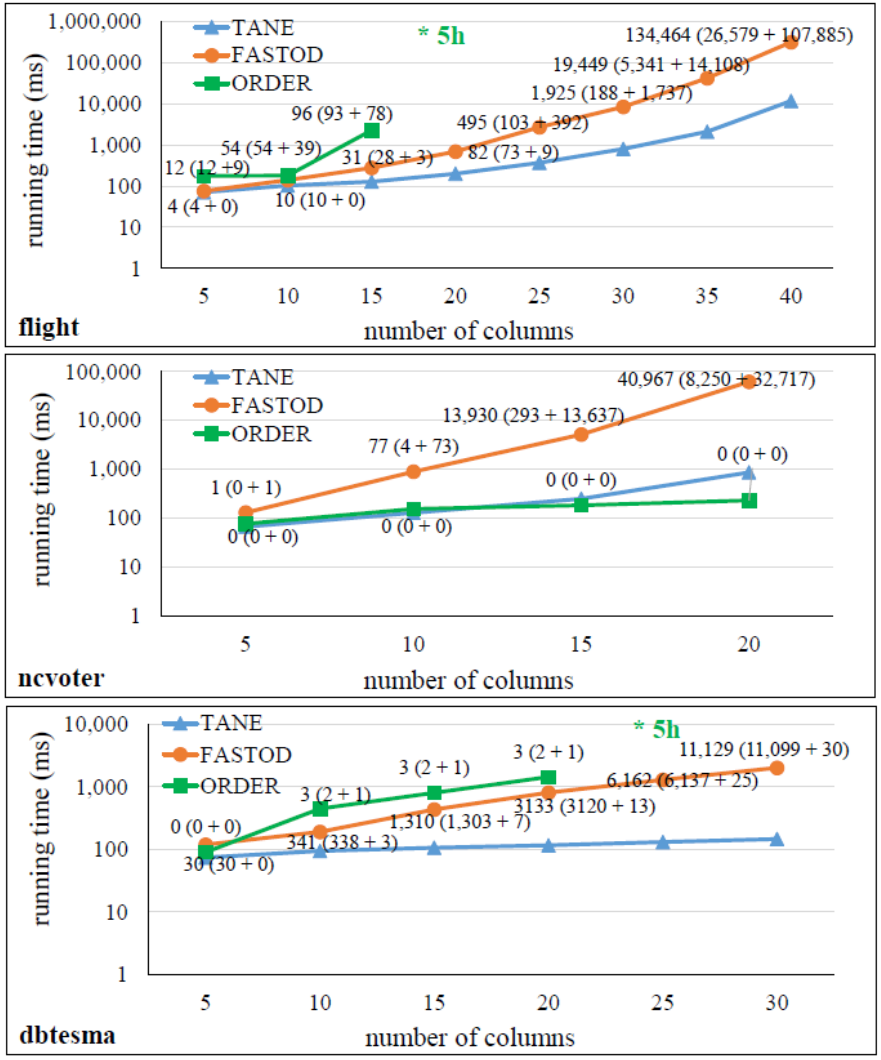
Experiments

- We compare our FASTOD algorithm with TANE for the discovery of FDs
- We also compare with ORDER, the only prior OD discovery algorithm
  - ORDER has a factorial worst-case time complexity in the # of attributes, and linear in the # of tuples
  - Our algorithm has an exponential worst-base time complexity in the # of attributes, and linear in the # of tuples

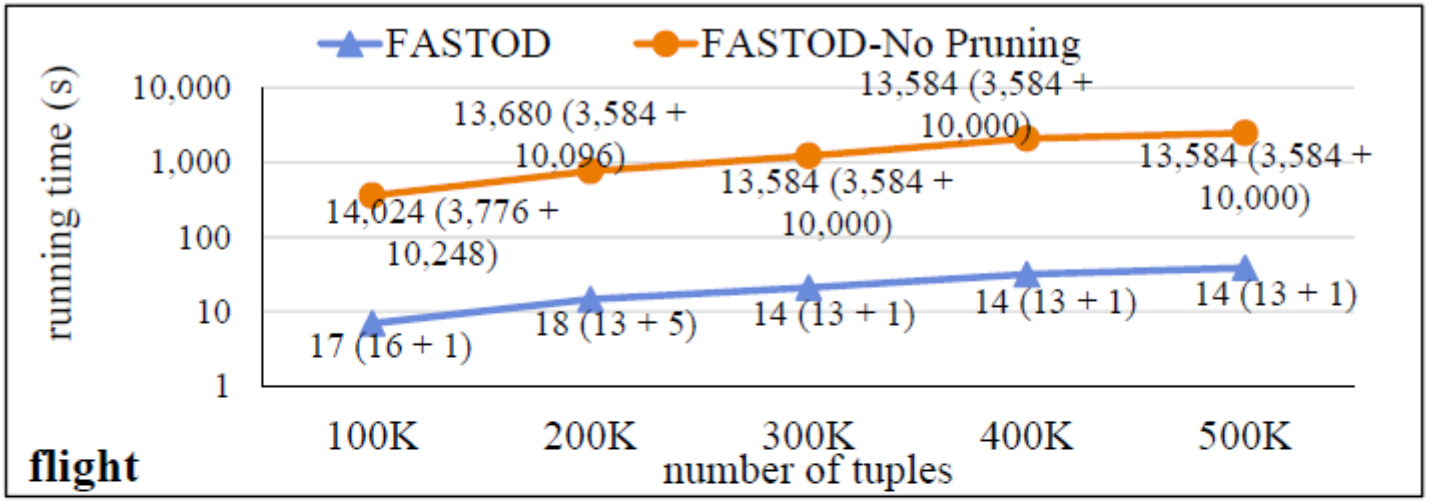
Scalability in # of Tuples

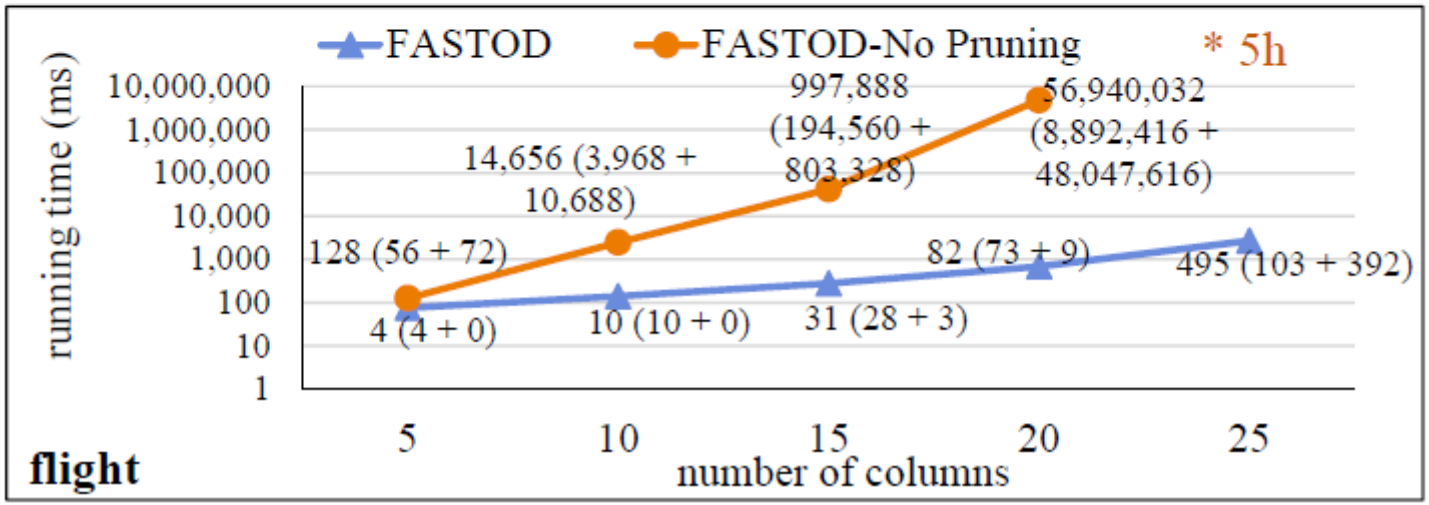


Scalability in # of Attributes



Effect of Pruning





Conclusions and Future Work

- We presented an efficient algorithm for discovering ODs
  - which we showed to be substantially faster than the prior-state-of-the-art
- The technical innovation that made our algorithm possible is:
  - a novel mapping into a set-based canonical form and
  - an axiomatization for set-based ODs
- Future work:
  - Bidirectional ODs (ascending and descending)
  - Conditional ODs that hold over portions of a relation