Deep and Collective Entity Resolution in Parallel

Ting Deng Wenfei Fan Ping Lu Xiaomeng Luo Xiaoke Zhu Wanhe An

Beihang University

University of Edinburgh

Shenzhen Institute of Computing Sciences

Deep and Collective Entity resolution (ER)

- Identify whether two tuples in a dataset refer to the same real-world entity
- It has been a longstanding challenge to improve the accuracy of ER

Deep and collective ER by unifying logic rules and ML models

Deep ER (Recursion)		*Collective ER*		*Unifying Logic and ML*
Identify new matches by *making use of matches deduced earlier*	+	Collectively correlate information *across multiple tables*	+	Benefit from both and improve the accuracy by *unifying the two methods*

Matching Rules with mL (MRLs)

MRL φ: $X \longrightarrow l$

Precondition X:
A conjunction of *predicates* over a database schema $\mathcal{R}$
$p ::= R(t) \mid t.A = c \mid t.A = s.B \mid \mathcal{M}(t[\bar{A}], s[\bar{B}])$

Consequence l:
An *id predicate* t.id$=s$.id or
an *ML predicate* $\mathcal{M}(t[\bar{A}], s[\bar{B}])$.

- Extend matching dependencies by embedding with *well-trained ML classifiers* as ML predicates
- t.id $= s$. id : the entities represented by t and s match, then refer to the same entity

Deep and Collective ER

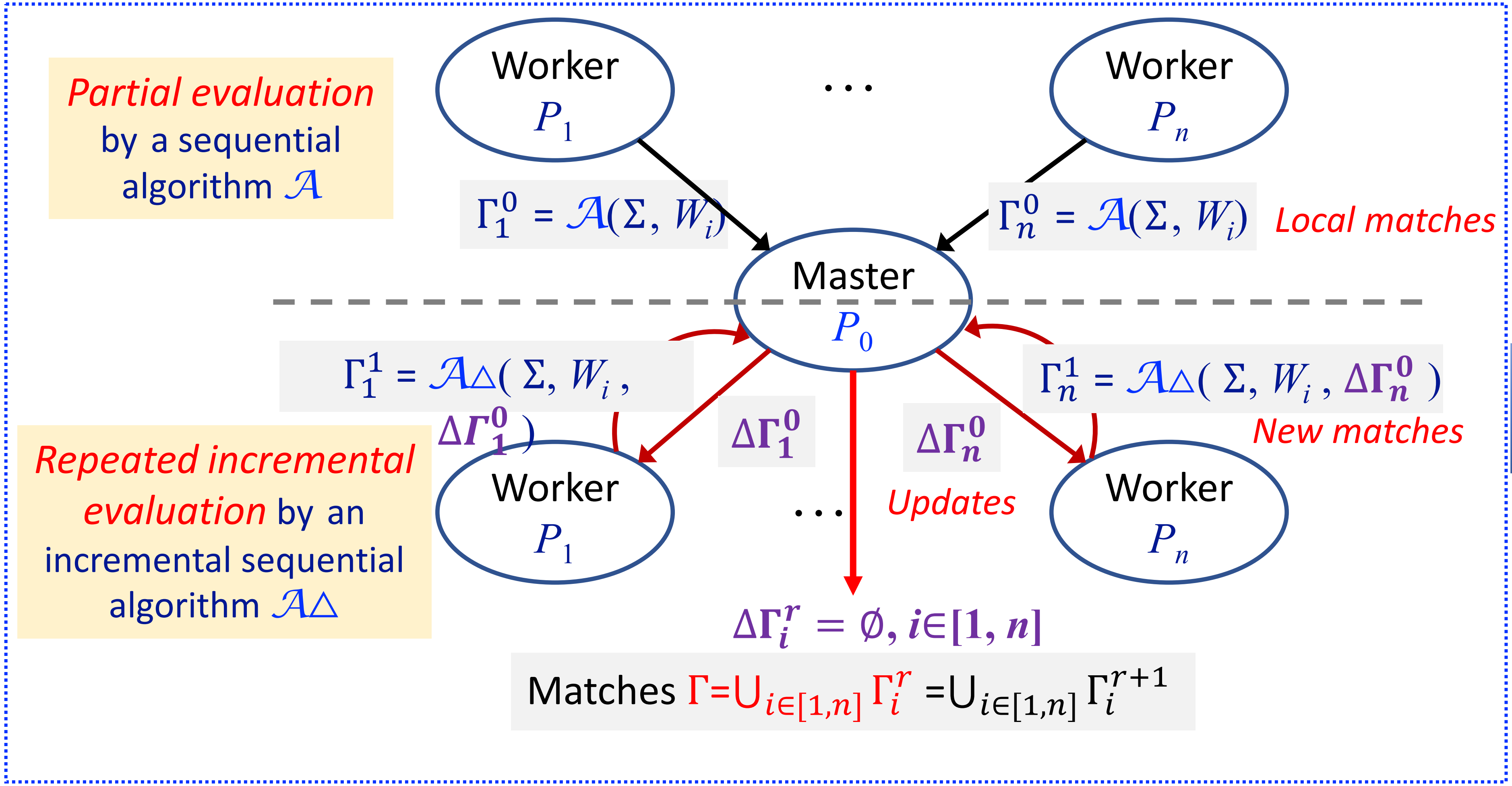
- Deduces a set Γ of *matches and validated ML predications* by applying MRLs in Σ
- Modeled as an extension of the *chase* with MRLs, with the *Church-Rosser* property

Decision problem	*Number of relations*	*id predicates* in the Prec.	
Deep and Collective ER	*Unbounded*	Yes	*NP-complete*
Collective ER (*not* Deep)	*Unbounded*	No	*NP-complete*
Deep ER (*not* collective)	Fixed	Yes	PTIME

Parallel fixpoint model

Updated-driven Parallel fixpoint model under BSP model

Partitioned dataset $\mathcal{D}= (W_1, \ldots, W_n)$, distributed to workers $P_1, \ldots, P_n$



Data partitioning strategy

Parallel deduction algorithm

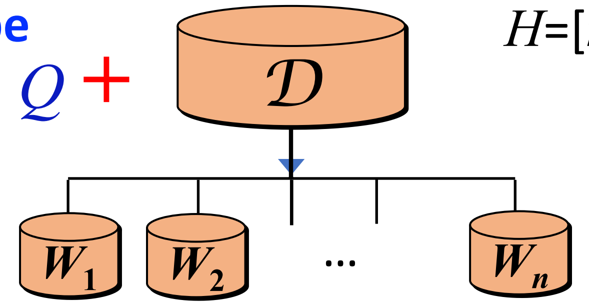
Update-driven fixpoint computation: no communication of raw data

Data partitioning for parallel fixpoint computation

Extend Hypercube (HC) to handle a set of MRLs with *multiple query optimization* (*MQO*)

Hypercube

Q +



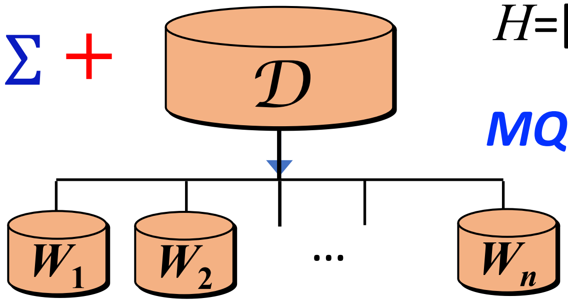
$H=[n_1]\times\cdots\times[n_l]$

A CQ query Q can be answered *locally*

$Q(\mathcal{D})=\bigcup_{i\in[1,n]} Q(W_i)$

HyPart

Σ +



$H=[n_1]\times\cdots\times[n_l]$

MQO

$\mathcal{D}\models\Sigma$ *can be verified locally*

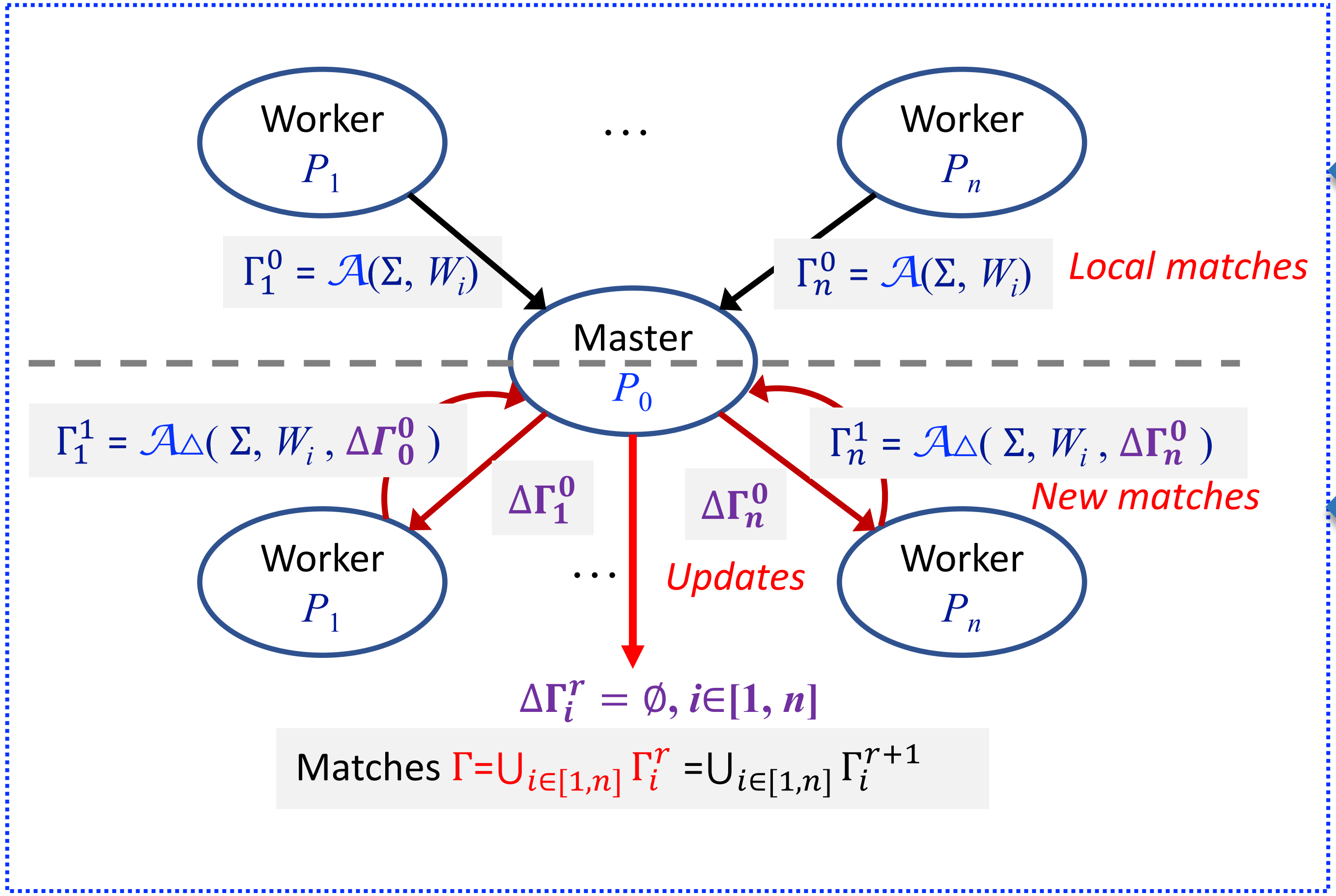
$\mathcal{D}\models\Sigma$ iff $W_i\models\Sigma$, $i\in[1,n]$

- Partition the data with the *minimum hash function computation* (*NP-complete*)

An *heuristic partitioning algorithm* HyPart */A strategy to assign the hash functions*

Reduce the *computation* cost	Reduce the *communication* cost	Improve the performance
Reuse the hash function computations as much as possible	Send a tuple with the same hash functions *to the same worker* for different rules	Using *MQO* to *find the common predicates* of MRLs

A Parallel scalable deduction algorithm



Partial evaluation by Deduce ($\mathcal{A}$)

Deducing matching in Γ *for each MRL and each of its valuations*

Repeated incremental evaluation by IncDeduce ($\mathcal{A}\triangle$)

Incrementally expand Γ by using an *update-driven* strategy

Optimization to speed up the process

Avoid to store all intermediate results and *reduce the repeated valuations*

Parallelly scalable: the more processors are used, the faster the algorithms run

Experimental results

- Using *five real-life Datasets* and synthetic datasets, compared with *eight state-of-the-art* baselines

Accuracy

- ***23% and 38% more accurate*** than ML and logical methods for ER, resp.
- outperforms deep ER and collective ER by 21% and 32%, resp.

Efficiency

- ***505s on datasets of 30M tuples*** using 16 machines.
- faster than 7 out of 8 state-of-the-art ER baselines.

Scalability

- ***Parallelly scalable*** with the number n of processors used: 3.56 times faster when n increases from 4 to 32

Models and Algorithms for Deep and Collective ER by unifying logic and ML