

Optimal Top-k Generation of Attribute Combinations based on Ranked Lists

Jiaheng Lu*, Pierre Senellart#, Chunbin Lin*, Xiaoyong Du*, Shan Wang* and Xinxing Chen*

* School of Information and DEKE, MOE, Renmin University of China

Institute Telecom; Telecom ParisTech, France

{jiahenglu, chunbinlin, duyong, swang, xxchen}@ruc.edu.cn, pierre@senellart.com



Motivation and Problem Statement

Forward		Center		Guard	
F1	F2	C1	C2	G1	G2
Juwan Howard	LeBron James	Chris Bosh	Eddy Curry	Dwanye Wade	Terrel Harris
(G01, 9.31)	(G02, 8.91)	(G05, 7.21)	(G01, 3.81)	(G02, 6.59)	(G09, 7.10)
(G07, 9.02)	(G08, 8.07)	(G02, 6.01)	(G06, 3.59)	(G03, 6.19)	(G03, 6.01)
(G03, 8.87)	(G05, 7.54)	(G06, 5.58)	(G04, 3.21)	(G04, 5.81)	(G04, 3.79)
(G04, 5.02)	(G10, 7.52)	(G10, 5.51)	(G07, 3.07)	(G05, 4.01)	(G08, 3.02)
(G11, 4.81)	(G03, 6.14)	(G04, 5.00)	(G09, 2.07)	(G01, 3.38)	(G05, 2.89)
(G08, 4.02)	(G01, 5.05)	(G11, 3.09)	(G11, 1.70)	(G09, 2.25)	(G02, 2.52)
(G06, 4.31)	(G04, 5.01)	(G01, 2.06)	(G10, 1.62)	(G06, 1.52)	(G01, 2.00)
(G05, 3.59)	(G09, 3.34)	(G08, 2.03)	(G02, 1.59)	(G08, 1.51)	(G10, 1.59)
(G09, 2.06)	(G06, 3.01)	(G09, 1.98)	(G08, 1.19)	(G07, 1.00)	(G06, 1.52)

Goal: select the **top-1** combination of athletes according to their best **top-2** aggregate scores for games where they played together.

F₂C₁G₁ is the best combination of athletes, as the top-2 games are G02 and G05, and 40.27 (= 21.51 + 18.76) is the highest overall score.

Problem Statement (Top-k,m)

Given a set of groups where each group contains multiple attributes and each attribute is associated with a ranked list.

Top-k,m returns **top-k combinations** of attributes which have the highest overall scores over their **top-m match instances** by a monotonic aggregate function.

(a) Source data of three groups

F1C1G1	F1C2G1	F1C1G2	F1C2G2	F2C1G1	F2C2G1	F2C1G2	F2C2G2
(G04, 15.83)	(G04, 13.81)	(G01, 16.50)	(G01, 15.12)	(G02, 21.51)	(G05, 17.64)	(G02, 17.09)	(G02, 13.02)
(G05, 13.69)	(G04, 14.04)	(G07, 12.05)	(G07, 12.05)	(G05, 18.76)	(G02, 17.44)	(G04, 14.03)	(G09, 12.51)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

(b) Top-2 aggregate scores for each combination

Essential difference between top-k queries and top-k,m problem

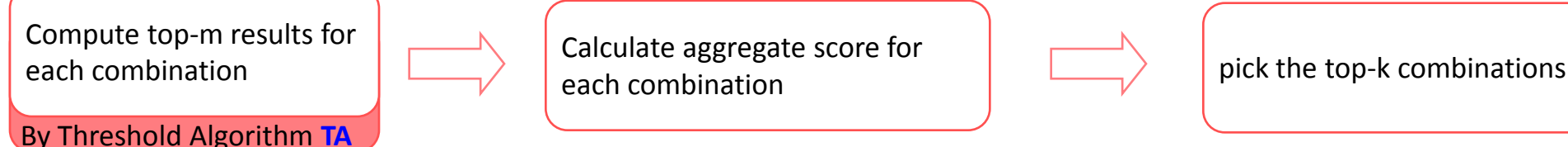
- Top-k query returns the **top-k tuples** (objects), while **Top-k,m** query returns the **top-k combinations** of attributes in groups.
- Top-k,m queries **cannot** be transformed into a SQL (nested) query, since SQL queries return tuples, while top-k,m returns attribute combinations based on ranked inverted lists.

Top-k,m Algorithms

Access model

- Sorted accesses:** read the tuple of lists sequentially.
- Random accesses:** quickly locate tuples whose ID has been seen by sorted access.

Baseline algorithm: ETA



Shortcoming

it needs to compute top-m results for each combination and reads more inputs than needed

ULA (Upper and Lower bounds Algorithm)



Upper Bound

$$e^{\min} = \begin{cases} F_2(\text{Score}(T_1^*), \dots, \text{Score}(T_m^*), T^*, \dots, T^*) & m^* < m \\ \max\{F_2(\text{Score}(T_1^*), \dots, \text{Score}(T_m^*))\} & m^* \geq m \end{cases}$$

Lower Bound

$$e^{\min} = \begin{cases} F_2(\text{Score}(T_1^*), \dots, \text{Score}(T_m^*), 0, \dots, 0) & m^* < m \\ \max\{F_2(\text{Score}(T_1^*), \dots, \text{Score}(T_m^*))\} & m^* \geq m \end{cases}$$

Hit-condition

One combination ϵ should be an answer if there are at least $N_{\text{com}} - k$ (N_{com} is the total number of the combinations) distinct combinations $\xi_1, \dots, \xi_{N_{\text{com}}-k}$ such that $e^{\min} \geq \max\{e_i^{\max} \mid 1 \leq i \leq N_{\text{com}} - k\}$

Although ULA is more efficient than ETA, ULA needs to compute the lower and upper bounds for each combination, which may be an expensive operation when the number of combinations is large.

Optimized top-k,m algorithm: ULA+

Pruning combinations without computing the bounds

Intuitively, the combinations with lists containing small top tuples are guaranteed not to be part of answers, as their scores are too small. Therefore, we **do not need to take time to compute their accurate upper and lower bounds**.

Domination

A combination $\epsilon = \{\epsilon_1, \dots, \epsilon_n\}$ is said to dominate another combination $\xi = \{\xi_1, \dots, \xi_n\}$ denoted $\epsilon \succeq \xi$ for every $1 < k < n$, either holds, $\epsilon_i = \xi_i$ or $L_{\epsilon_i} > L_{\xi_i}$ are two (possibly identical) attributes of the same group G_i .

if ϵ dominates ξ then the upper bound of ϵ 's greater than or equal to that of ξ .

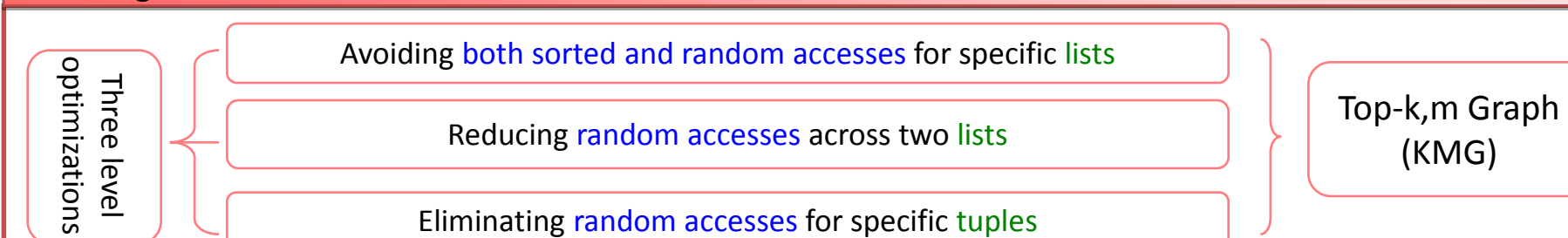


Example

A1	A2	A3	A4	B1	B2	B3
(a, 10)	(c, 8.3)	(a, 6.3)	(c, 4.8)	(b, 9.0)	(e, 8.0)	(a, 5.8)
(b, 5.8)	(d, 7.1)	(d, 3.7)	(d, 4.3)	(a, 8.2)	(f, 3.2)	(d, 4.5)

1. find the seed combination $\xi = (A_2, B_1)$
 2. Drop useless combinations: All combinations (A_i, B_j) ($\forall i \in [3, n], j \in [2, n]$), as well as (A_2, B_j) and (B_1, A_i) are dominated by ξ and can be pruned quickly.

Reducing the number of accesses

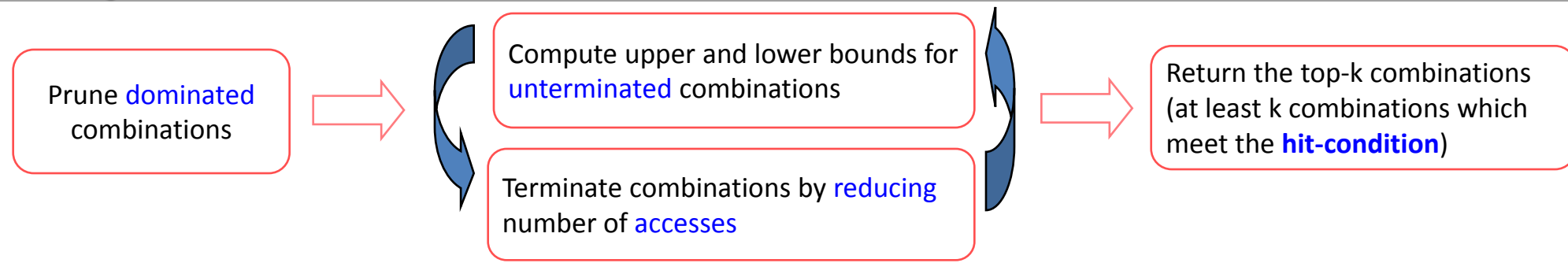


Operations in KMG

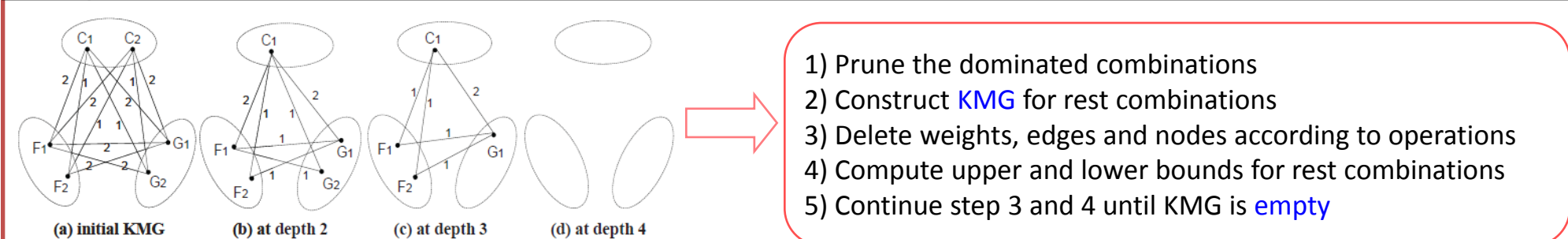
- decreasing the weight of edges by 1 if one of combinations involving the edge is terminated
- deleting the edge if its weight is 0, which means that random accesses between the two lists are useless
- removing the node if its degree is 0, which indicates that both sorted and random accesses in this list are useless

Optimized top-k,m algorithm: ULA+

ULA+ Algorithm



Example



Optimality properties

Instance Optimality

- Let D be the class of all databases. Let A be the class of all algorithms that correctly find top-k,m answers for every database and that **do not make wild guesses**. If the size of each group is treated as a constant, then ULA and ULA+ are **instance-optimal** over A and D.

The upper bound of the optimality ratio is tight

- There is **no** deterministic algorithm that is instance-optimal for top-k,m problem, with optimality ratio **less than** $T + K C_r / C_s$, where $T = \sum_{i=1}^n g_i$, $K = \sum_{i \neq j} (g_i g_j)$

No Instance Optimal Algorithms

- Let D be the class of all databases. Let A be the class of all algorithms (**wild guesses are allowed**) that correctly find top-k,m answers for every database. There is **no** deterministic algorithm that is **instance-optimal** over A and D.

XML Keyword Refinement

Example

- Given a query $Q = \{DB; UC Irvine; 2002\}$
- Groups $G_1 = \{DB; UC Irvine\}$, $G_2 = \{UC Irvine; 2002\}$, and $G_3 = \{2002\}$.

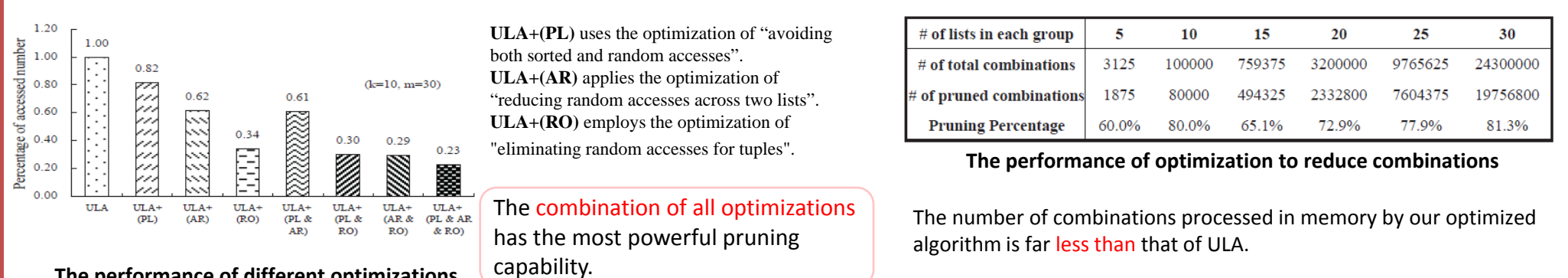
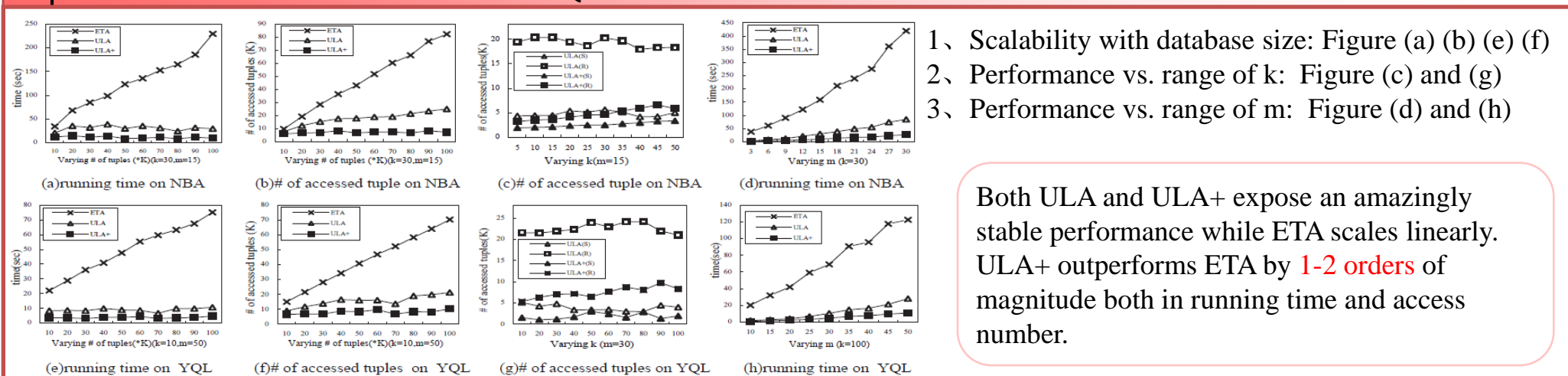
Consider a **top-1,2** query, 17 combinations in total.

$Q' = \{DB, UC Irvine, 2002\}$ with two SLCA nodes (i.e., 1.3.10 and 1.2) The lower bound of Q' is $0.567 + 0.729 = 1.296$

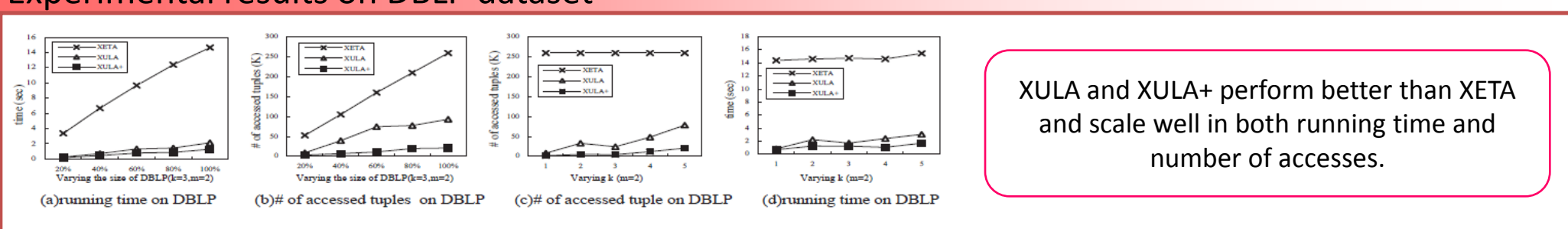
Experimental Study

Experimental Setup and Data Sets				Metrics			
Language: Java; OS: Windows XP; CPU: 2.0GHz; Disk:320GB							
Data set	# of objects	# of groups	group size	# of combinations	(1) running time	(2) access number	(3) number of processed combinations
YQL	100,100	3	3	150	12	3,375,000	1,728
NBA	31,200	5	5	32	6	33,554,432	7,776
DBLP	3,786,406	7	2.6	12	5	371,292	327

Experimental results on NBA and YQL datasets



Experimental results on DBLP dataset



Conclusion

- Propose a new problem called **top-k,m** query evaluation
- Developed a family of efficient algorithms, including **ULA** and **ULA+**
- Analyzed different classes of data and access models, such as group size and wild guesses and their implication on the **optimality** of query evaluation
- Showed how to adapt our algorithms to the context of **XML** keyword query **refinement**