### Motivation and Problem Statement

**Top-k,m Algorithms**

**Access model**
- Sorted accesses: read the tuple of lists sequentially.
- Random accesses: quickly locate tuples whose IDs have been seen by sorted access.

**Baseline algorithm: ETA**

- Compute top-m results for each combination.
- Calculate aggregate score for each combination.
- Pick the top-k combinations.

**ULP (Upper and Lower Bounds Algorithm)**

- Compute the upper and lower bounds for each combination.
- Return the top-k combinations that meet the bit-condition.

Although ULP is more efficient than ETA, ETA needs to compute the upper and lower 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.

**Dimension**

A combination \(C = (A_1, B_1, G_1, \ldots, A_n, B_n, G_n)\) is said to dominate another combination \(C' = (A'_1, B'_1, G'_1, \ldots, A'_n, B'_n, G'_n)\) if for every \(1 \leq i \leq n\), either holds, \(a_i \leq a'_i\) or \(A_i = B_i\), or two (possibly identical) attributes of the same group \(G_i\) either holds, \(G_i = G'_i\) or \(G_i \not\subseteq G'_i\).

If \(d\) dominates \(C\), then the upper bound of \(C\) is greater than or equal to that of \(d\).

**Seed combination selection**

1. Find the seed combination \(d = (A, B, G)\) such that \(a_{min} \in A\) and \(b_{min} \in B\).
2. Drop useless combinations: All combinations \((A_i, B_i, G_i)\) such that \(A_i \not\subseteq A\) or \(B_i \not\subseteq B\) are dominated.

**Rules of XUL**

- **XUL (with lower bound)**
  - XUL can optimally prune combinations by computing a lower bound (LB) for each combination, \(LB(C)\).
  - \(LB(C)\) is a function of the attributes in \(C\), \(LB(C) = f(A, B, G)\).

**Experimental Study**

**Metrics**

- **Scalability with database size:** Figure (a) (b) (c) (d)
- **Performance vs. range of \(k\):** Figure (e) (f) (g)
- **Performance vs. range of \(m\):** Figure (d) (h)

Both ULA and ULA+ expose an amazingly stable performance while a 20% scale linearly. ULA+ outperforms ETA by 6-2 orders of magnitude in both running time and access number.

**Conclusion**

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