Charu C. Aggarwal[1], Yan Li[2], Jianyong Wang[2], Jing Wang[3]
1. IBM T J Watson Research Center
2. Tsinghua University
3. New York University

Frequent Pattern Mining with Uncertain Data

ACM KDD Conference, 2009

Introduction

- Uncertainty is everywhere
  - Errors in Instrumentation
  - Derived data sets
  - Links between privacy and uncertain data mining
    * Intentionally incorporated uncertainty
- The results of data mining algorithms are highly impacted by the uncertainty
- The frequent pattern mining problem is one for which the performance is significantly impacted by uncertain representations

Problem Definition for Uncertain Data

- Associate *existential probabilities* for items in transactions
- Probability of presence of item i in transaction T_k is $p(i, T_k)$.
- Expected support of itemset I in T_k is $p(I, T_k) = \pi_{i \in I} p(i, T_k)$
- Expected support of itemset I is $\sum_k p(I, T_k)$
- **Definition:** Determine all frequent patterns with expected support above user-defined threshold

Deterministic Algorithm Classes

- Candidate Generate-and-Test Algorithms
  - Join based
  - Tree based

- Pattern Growth Algorithms
  - H-Mine
  - FP-Growth

Contributions

- Discuss extensions of broad classes of frequent pattern mining algorithms
- Compare the broad classes of frequent pattern mining algorithms
- Stress-test on computationally difficult case: high uncertainty probabilities
- Memory is an important resource in the uncertain case: test for memory requirements

Key Take Aways

- Algorithms which work well on deterministic data (FP-Growth) may not work as well on uncertain data

- Pruning tricks which work for low uncertainty probabilities are an overhead for the case of high uncertainty probabilities

- The pattern-growth paradigm can be leveraged if it is used in the proper context

  - Extensions of the H-mine algorithm turn out to be the most effective in terms of the combination of memory and computational requirements

Apriori Extensions

- Standard candidate-generate-and-test can be extended directly with the main difference being in counting

- Chiu et al proposed several pruning techniques
  - **Transaction Trimming Methods:** Key is in pruning infrequent items
  - **Support Pruning Methods:** Compute upper bounds on expected support of itemsets; prune when they fall below minimum support

Tree Based Generate-and-Test Algorithms

- Tree based algorithms generate a trie of candidate itemsets
- Can directly be generalized to the uncertain case
- Pruning conditions for deterministic case hold for uncertain case
- Projected databases can be constructed as in deterministic case, except that uncertainty probabilities also need to be maintained

The FP-Tree Technique: Challenges

- The FP-Tree technique generates a *compressed representation* of the database by *sharing information* about prefixes

- **Uncertain Challenge:** The prefixes contain information about probabilities which is *specific to each transaction*.

  – Implies that effective sharing is not possible

Straightforward Solution

- Treat each distinct probability as a separate node (no sharing between two transactions with the same item but distinct probabilities) (Leung et al)

- Criticism:
  - Effective only if a lot of items have exactly the same distinct probability
  - Otherwise compression of FP-Tree is not good, and leads to too much overhead
  - In continuous domain of probability, the assumption of exactly the same probability value is not reasonable

Our Solution

- Create cluster ranges of probabilities
- Construct a node for each clustered range (allows some node sharing)
- Use FP-Tree algorithm to generate a close *superset* of the frequent itemsets
  - **Key:** Prove upper bound property of expected supports
- Remove irrelevant itemsets in a final pass

Two Variants

- *UFP-growth algorithm:* Adopts the recursively pattern-growth method used in FP-growth

- *UCFP-growth algorithm:* Constructs only the conditional FP-Tree for each frequent item *at the first level* and mines frequent itemsets for each conditional tree.

Observations

- Key selling point of FP-Tree is transaction database compression by information sharing: not effective in the uncertain environment

- Another selling point is the use of the pattern growth paradigm

- Is it possible to leverage the pattern-growth paradigm without worrying about the node sharing issue of FP-Tree?
  - **Solution:** Extend H-Mine

Uncertain Extension of H-Mine

- The H-mine structure uses the linkage behavior among transactions corresponding to a branch of the FP-Tree without actually creating a projected database

- **Uncertain Extension:** Maintains item probabilities in original database, and uses linkage behavior to traverse database efficiently

- Prefix probabilities can be computed on the fly by using the information associated with original transaction

Observations (UH-Mine)

- Overall Effect: Uses the linkages to effectively traverse the transaction set without worrying about information sharing of the FP-Tree

- This approach is better than FP-Tree even in the deterministic case, when compression from FP-Tree is not high

- This will turn out to be particularly true for the uncertain case

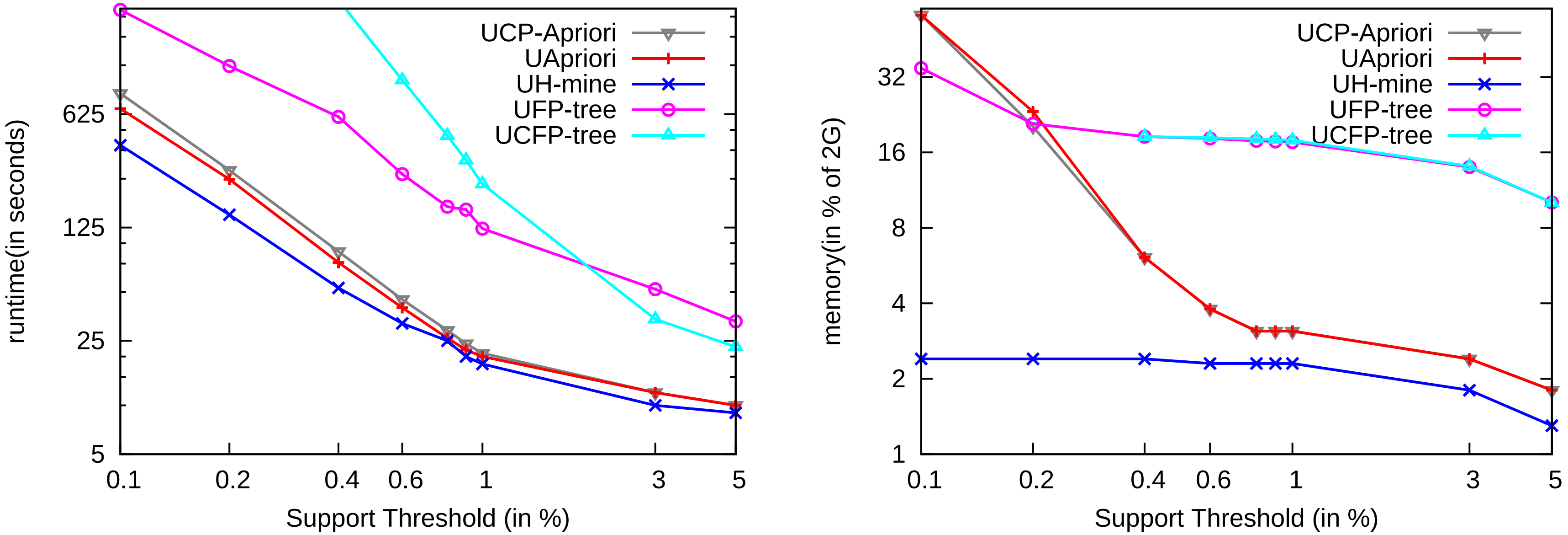
Experimental Results

- Use Connect4, kosarak, and T40.I10.D100K

- Generate dense uncertainty probabilities

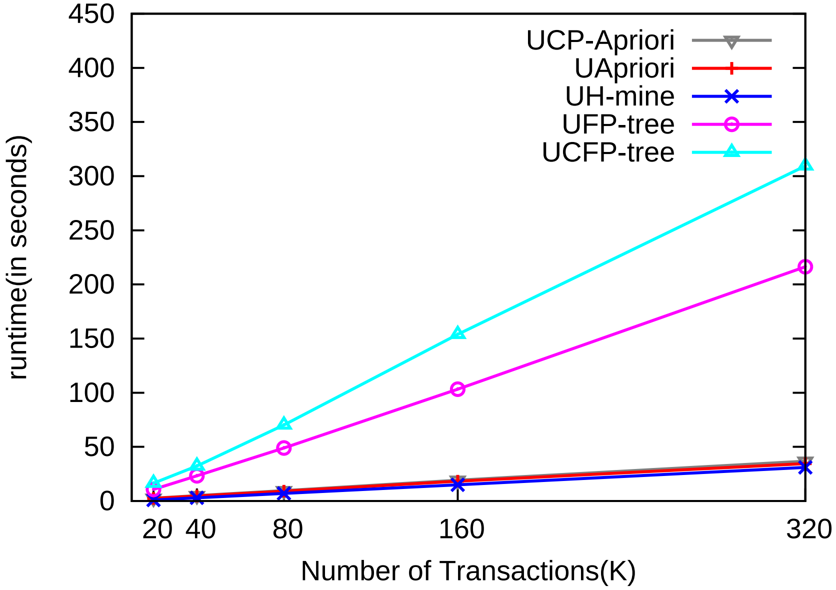
- Difficult case where rapid fall off in probabilities with increasing pattern length is not available

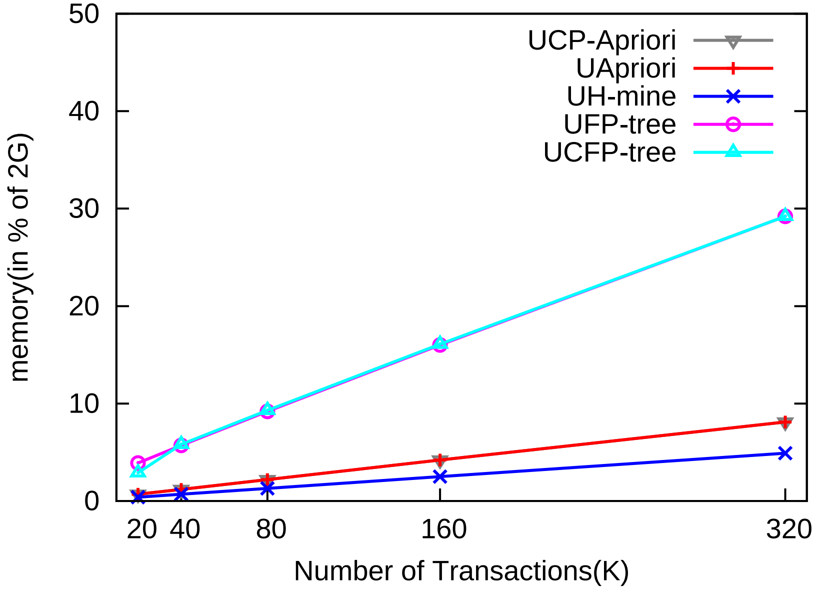
T40.I10.D100K



- Running Time and Memory Requirements

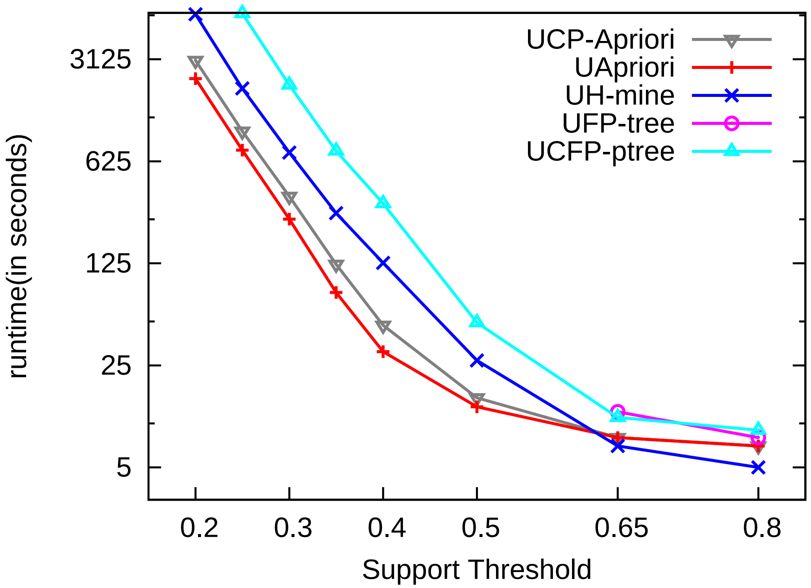
Scalability

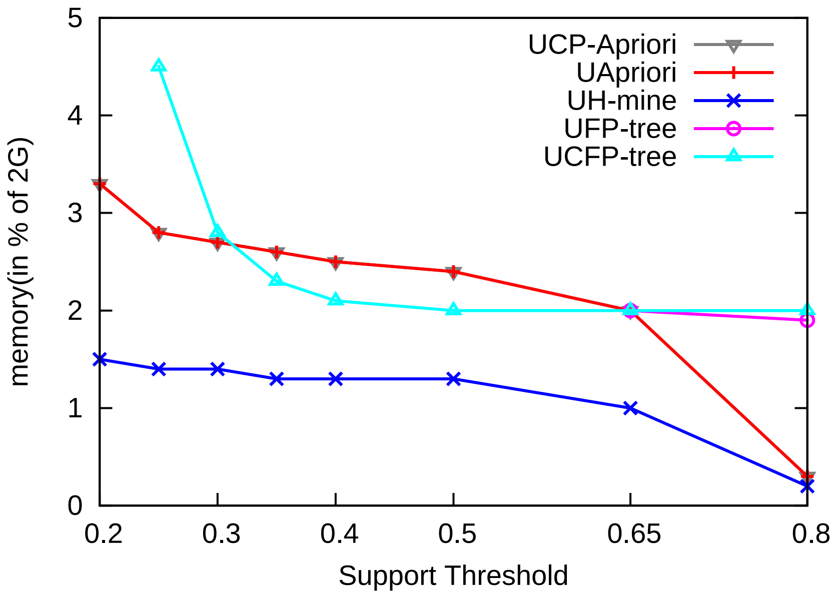




- Running Time and Memory Requirements with increasing number of transactions

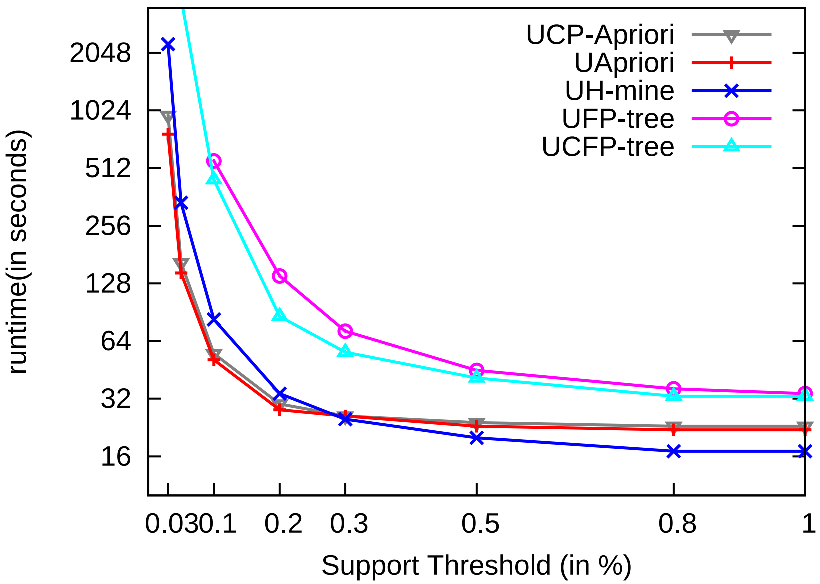
Connect4

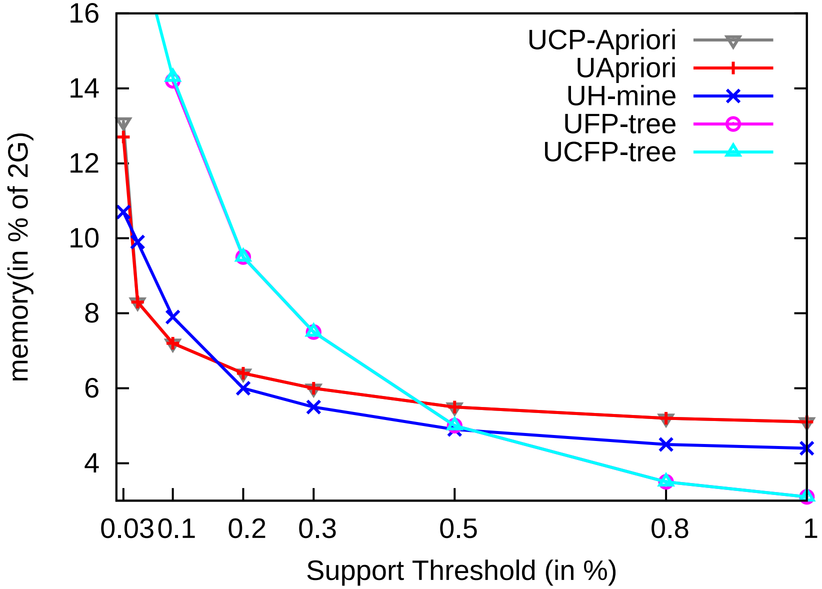




- Running Time and Memory Requirements

Kosarak





- Running Time and Memory Requirements

Conclusions

- Algorithms which work well on deterministic data (FP-Growth) may not work as well on uncertain data

- Pruning tricks which work for low uncertainty probabilities are an overhead for the case of high uncertainty probabilities

- Extensions of the H-mine algorithm turn out to be the most effective in terms of the combination of memory and computational requirements