Reductions for Frequency-Based Data Mining Problems

Stefan Neumann & Pauli Miettinen
Maximal Frequent Patterns

- A **pattern** is a subset of the data entities
  - itemset, subgraph, subsequence, ...
- A pattern is **frequent** if it appears sufficiently often in the data
- A frequent pattern is **maximal** if it is not contained in any other frequent pattern
- Studied since 1990s
Computational Complexity

• Comp. complexity of maximal pattern mining surprisingly unknown
  • Potentially exponentially many max. patterns
    ⇒ takes exponential time

• More fine-grained answers:
  • Time w.r.t. input and output
    (enumeration complexity, Johnson et al. 1988)
  • Time spent to count the number of maximal patterns
    (counting complexity, Valiant 1979)
Reductions

• A can be reduced to B if we can solve A effectively with an algorithm to solve B
  • ”B is at least as hard as A”

• In this talk: maximality-preserving reductions between frequent pattern mining problems
  • ”Maximum X mining is at least as hard as maximum Y mining”
State of the Art

Uniquely labelled undirected graphs

MaxFS\((\text{BDG}^{3})\)

MaxFS\((\text{G})\)

MaxFS\((\text{PLN})\)

MaxFS\((\text{BTW}^{3})\)

MaxFIS

MaxSQS

MaxFS\((\text{T})\)

MaxFS\((\text{DAG})\)

Directed cyclic graphs

Directed graphs

Sequences with no repetition

Undir. trees

Planar undir. graphs

Undir. graphs with treewidth ≤ 3

Undir. graphs with degree ≤ 3

\(A \rightarrow B = A\) can be reduced to \(B\)
Maximality-Preserving Reductions

MaxFS($BDG^3$) $\rightarrow$ MaxFS($BTW^3$) $\rightarrow$ MaxFS($PLN$) $\rightarrow$ MaxFS($T$) $\rightarrow$ MaxFIS $\rightarrow$ MaxSQS $\rightarrow$ MaxFS($DAG$) $\rightarrow$ MaxFS($DirG$)

These reductions preserve enumeration and counting complexity

$A \rightarrow B = A$ can be reduced to $B$
Impressed?

• Why no more reductions?

• Example: From MaxFS(\textbf{G}) to MaxFIS
  
  • Each edge \{u, v\} has a unique label \((l(u), l(v))\)
  
  • Make the edges as items and graphs as transactions

  • Mine maximal frequent itemsets

• This doesn’t (quite) work!
What’s Wrong?

Frequent itemsets (minfreq 2/3):

- C D (3)
- A B (2)
- B C (2)
- B C D (2)

Not connected!
Feasible Patterns

- To be able to encode the connectedness, we need to constraint the feasible patterns
- We can adjust our reductions to work with these constraints. E.g.:
  - maximal graph patterns must map to maximal feasible itemsets, and
  - it must be easy to compute the graph patterns from the feasible maximum itemsets
- These constraints are transitive
Maximality-Preserving Reductions for Feasible Patterns

The complexity collapses under these reductions!

\[ A \rightarrow B = A \text{ can be reduced to } B \]
Maximality-Preserving
Reductions for Feasible Patterns

The complexity collapses under these reductions!

A → B = A can be reduced to B
Summary

• For all feasible pattern versions of the problems:
  • Enumerating all feasible patterns is \#P-hard
  • Given a set of feasible patterns, deciding whether there is any more feasible patterns is NP-hard
    • Even if only two patterns are given
  • For any fixed minfreq threshold \( \tau \), the enumeration can be done in polynomial time
Conclusions

- Most maximal pattern mining problems are essentially equally hard
  - Methods for one type of problem can be used to solve other types, as well
  - Feasible patterns admit usually constraints that are amenable to standard level-wise algorithms
- Notable exceptions: MaxFS on general graphs and sequences with repetitions
  - Subgraph isomorphism is NP-hard

Thank You!