FINDING TOPOLOGICAL FREQUENT PATTERNS FROM GRAPH DATASETS

Karel Vaculík

Assumptions on graphs

- Undirected
- Connected
- Labeled vertices and edges (not necessarily uniquely)

Graph isomorphism

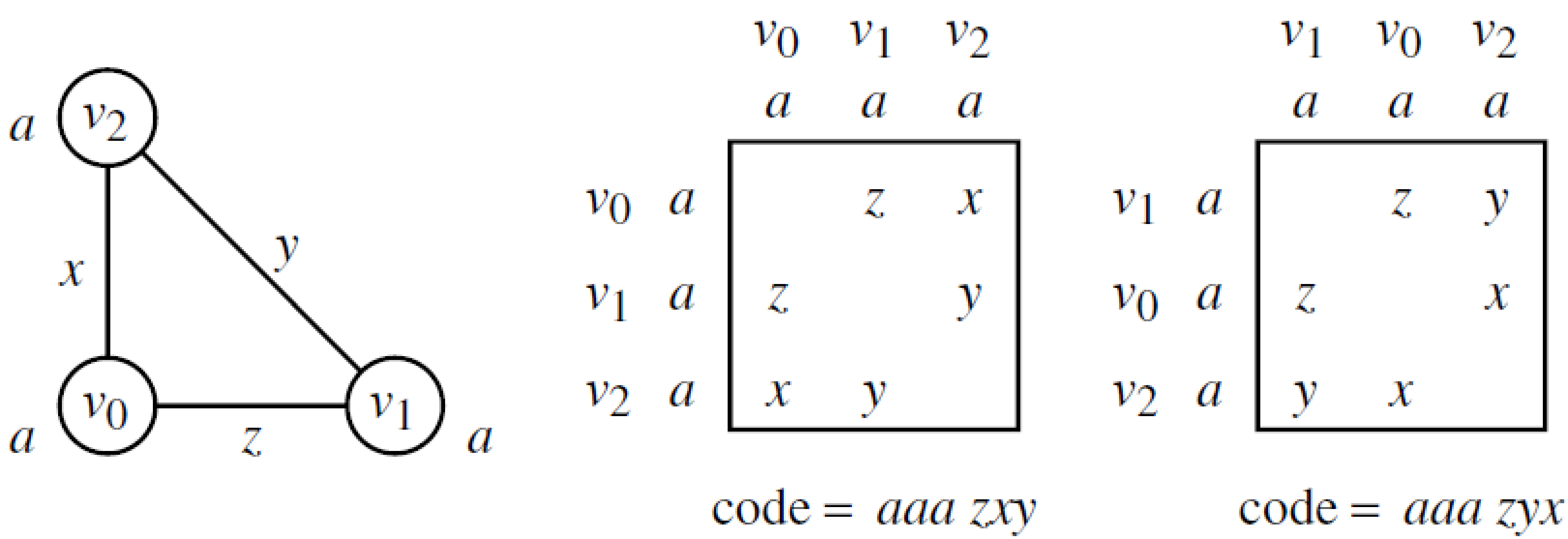
- Subgraph isomorphism
  - Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, find an isomorphism between G_2 and a subgraph of G_1 (≈ determine whether or not G_2 is included in G_1)

Graph isomorphism

- Canonical labeling
  - Unique code for the set of graphs with the same topological structure and the same labeling
  - In most cases:
    - Flattened representation of the adjacency matrix
    - Try all permutations in order to find minimum (or maximum) according to lexicographic ordering

Graph isomorphism

- Simple examples of codes and canonical adjacency matrices (from FSG):



(a) G^3 (b) (c)

Graph isomorphism

- Both problems are not known to be either in P or in NP-complete
  - In practice: complexity of canonical labeling is reduced by using various heuristics and properties in set of graphs

Two forms of the input

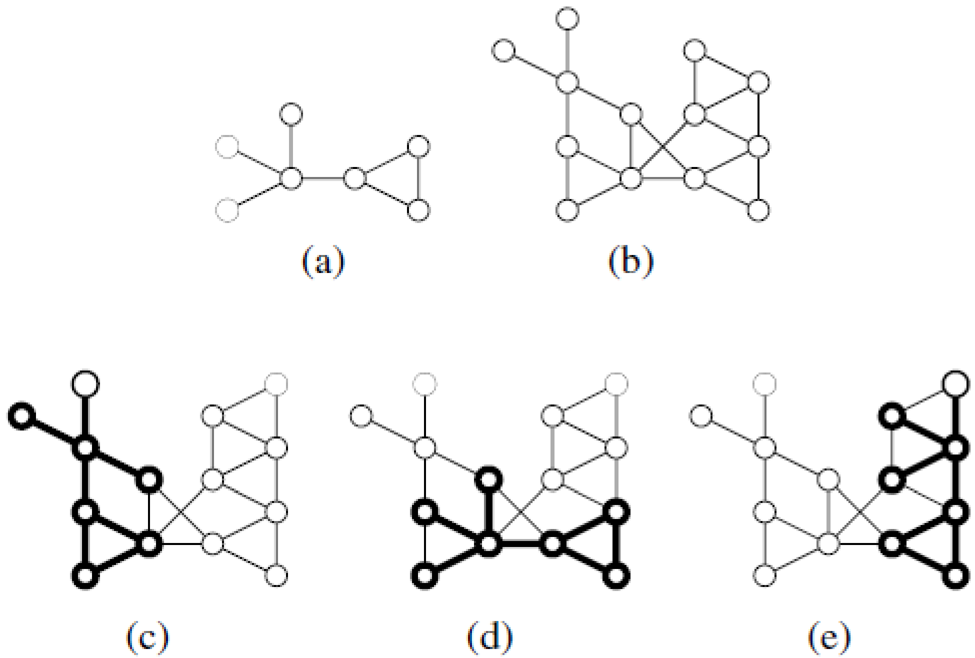
- *Graph-transaction setting*
  - Set of relatively small graphs (*transactions*)
  - Frequency (of a pattern): number of graph transactions that the pattern occurs in

Two forms of the input

- *Single-graph setting*
  - One large graph
  - Frequency: number of pattern occurrences in the single graph.
  - Counting the frequency of edge-disjoint embeddings (using *maximum independent set*)
  - Algorithms can be adapted to solve the first group
  - Examples of algorithms: SUBDUE, SEuS, GREW, SIGRAM, GBI
  - Not discussed further

Two forms of the input

Example:



Overlapped embeddings: (a) subgraph, (b) input graph, (c) embedding 1, (d) embedding 2, (e) embedding 3

Completeness of algorithms

- Complete algorithms
  - All frequent patterns that satisfy a given specification (e.g. minimum support threshold)
  - May become unfeasible
- Heuristic algorithms
  - Return only a subset of all frequent patterns (approximate solution)

Apriori-based algorithms

- FSG [3]
  - Complete
  - Transactional setting, connected graphs
  - Level-by-level expansion as in Apriori
  - Features:
    1. Sparse graph representation
    2. Adding one edge at a time
    3. Using canonical labeling and graph isomorphism
    4. Scales (linearly) with the database size

Apriori-based algorithms

- FFSM (Fast Frequent Subgraph Mining) [4]
  - Canonical form: appended columns from adjacency matrix
  - Competitive with gSpan
  - Algebraic graph framework

Apriori-based algorithms

- AGM (Apriori-based Graph Mining) [5]
  - Code of Adjacency Matrix

$$X_k = \begin{pmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \cdots & x_{1,k} \\ x_{2,1} & x_{2,2} & x_{2,3} & \cdots & x_{2,k} \\ x_{3,1} & x_{3,2} & x_{3,3} & \cdots & x_{3,k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{k,1} & x_{k,2} & x_{k,3} & \cdots & x_{k,k} \end{pmatrix}$$

$$code(X_k) = x_{1,1}x_{1,2}x_{2,2}x_{1,3}x_{2,3}x_{3,3}x_{1,4}\cdots x_{k-1,k}x_{k,k},$$

- AcGM [6]
    - Complete search of frequent connected (induced) subgraphs in a massive labeled graph dataset within highly practical time

Apriori-based algorithms

- Warmr [7]
  - ILP data mining algorithm
  - Datalog to represent both data and patterns
  - Patterns can reflect one-to-many and many-to-many relationships

Pattern growth algorithms

- gSpan (graph-based Substructure pattern mining) [8]
  - Without candidate generation
  - Adopts depth-first search strategy to mine frequent connected subgraphs
  - Canonical labeling based on DFS traversing of graph
  - Outperforms FSG

Pattern growth algorithms

- GASTON (GrAph/Sequence/Tree extractiON) [9]
  - "Quickstart principle"
    - Various substructures are contained in each other
    - First consider paths, then transform them to trees and finally transform trees to graphs
    - More efficient algorithms for simple substructures, advanced algorithms only when really needed
  - Observation: most frequent substructures in practical graph databases are free trees (trees with no vertex designated as a root)

References

1. Diane J. Cook, Lawrence B. Holder. *Mining graph data*. John Wiley and Sons, 2007.
2. Karsten Borgwardt and Xifeng Yan. *GRAPH MINING*. http://agbs.kyb.tuebingen.mpg.de/wikis/bg/BNA-4.pdf
3. M. Kuramochi and G. Karypis. *Frequent subgraph discovery*. In Proceedings of 2001 IEEE International Conference on Data Mining (ICDM), pp. 313–320, November 2001.

References

4. J. Huan, W. Wang, and J. Prins. *Efficient mining of frequent subgraph in the presence of isomophism*. In Proceedings of 2003 IEEE International Conference on Data Mining (ICDM'03), pp. 549–552, 2003.
5. A. Inokuchi, T. Washio, and H. Motoda. *An apriori-based algorithm for mining frequent substructures from graph data*. In Proceedings of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'00), pp. 13–23, Lyon, France, September, 2000.

References

6. A. Inokuchi, T. Washio, K. Nishimura, and H. Motoda. *A fast algorithm for mining frequent connected subgraphs*. Technical Report RT0448, IBM Research, Tokyo Research Laboratory, 2002.
7. Ross D. Kinga, Ashwin Srinivasanb & Luc Dehaspec. *Warmr: a data mining tool for chemical data*. Journal of Computer-Aided Molecular Design, Volume 15, Issue 2, pp 173-181. Kluwer Academic Publishers, 2001.

References

8. X. Yan and J. Han. *gSpan: Graph-based substructure pattern mining*. In Proceedings of 2002 IEEE International Conference on Data Mining (ICDM), pp. 721–724, 2002.
9. S. Nijssen and J. N. Kok. *A quickstart in frequent structure mining can make a difference*. In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), pp. 647–652, 2004.