

Mining Minimal Contrast Subgraph Patterns

Roger Ming Hieng Ting*

James Bailey*

Abstract

In this paper, we introduce a new type of contrast pattern, the minimal contrast subgraph. It is able to capture structural differences between any two collections of graphs and can be useful in chemical compound comparison and building graph classification models. However, mining minimal contrast subgraphs is a challenging task, due to the exponentially large search space and graph (sub)isomorphism problems. We present an algorithm which utilises a backtracking tree to first compute the maximal common edge sets and then uses a minimal hypergraph transversal algorithm, to derive the set of minimal contrast subgraphs. An experimental evaluation demonstrates the potential of our technique for finding interesting differences in graph data.

Keywords: Graph mining, hypergraph transversals

1 Introduction

In this paper, we introduce a new type of pattern for contrasting collections of graphs, called a *minimal contrast subgraph*. A contrast subgraph is essentially a subgraph appearing in one class of graphs, but never in another class of graphs. It is minimal if none of its subgraphs are contrasts. There are many situations where minimal contrast subgraphs can be applied, such as comparing structural differences between chemical compounds. There is a large body of previous work on graph comparison, focusing on problems such as subgraph isomorphism, graph isomorphism, common subgraph and maximum common subgraph [14, 10, 7, 11]. However, these approaches have their limitations. Unlike minimal contrast subgraphs, they cannot be directly used to enumerate interesting differences between classes of graphs. They are unable to answer questions like “What are the smallest structural differences between two chemical compounds?” or “How did the network topology change over a period of time?”. Also, we believe minimal contrast subgraphs have strong potential for building classifiers.

Several challenges arise in the mining of minimal contrast subgraphs. Firstly, it involves matching operations which are computationally demanding. Indeed,

*NICTA Victoria Laboratory, Department of Computer Science and Software Engineering, University of Melbourne, Australia.

just testing whether a graph is a subgraph of another is NP-complete. Secondly, we are interested in discovering the complete set of contrast subgraphs. This is unlike many frequent subgraph mining algorithms, which focus on mining only connected subgraphs or induced subgraphs and omit subgraphs that are isomorphic [5, 8, 15, 2]. We believe allowing disconnected subgraphs is important, since it facilitates a very succinct representation of differences, yet is still useful for domain experts. However, allowing disconnectedness has a price, since it makes the search space of possible contrasts larger. Therefore, a key focus of this paper is to investigate the limits of contrast mining under this difficult requirement. We make the following two important contributions: i) We introduce the concept of minimal contrast subgraphs, formally clarify the relationships between contrast subgraphs and the associated notion of contrast edge sets and identify the connection between minimal contrast edge sets and the dual notion of maximal common edge sets, using the technical device of hypergraph transversals, ii) We present an algorithm for mining minimal contrast subgraphs and experimentally show its potential for finding interesting differences.

2 Preliminary Concepts

We now provide some necessary definitions and also give a new result that links together minimal contrast subgraphs and two other types of graphs: minimal contrast vertex sets and minimal contrast edge sets. The focus of this paper will be on finding minimal contrast subgraphs between a single positive graph and a set of negative graphs. All graphs are assumed to be undirected, labelled, simple graphs. The positive graph is labelled G_p and each negative graph as G_n or G_{n_i} .

DEFINITION 2.1. A **labelled graph** G is a 4-tuple (V, E, α, β) , V is a vertex set, $E \subseteq V \times V$ is a set of edges, α is a function assigning labels to vertices and β is a function assigning labels to edges.

DEFINITION 2.2. $S = (W, F, \alpha, \beta)$ is the **subgraph** of $G = (V, E, \alpha, \beta)$ iff (1) $W \subseteq V$ and (2) $F \subseteq E \cap (W \times W)$. Equivalently, G is said to be the **supergraph** of S .

$S \subseteq G$ indicates that S is the subgraph of G and \subset is used to indicate strict inclusion.

DEFINITION 2.3. $I = (W, F, \alpha, \beta)$ is the **induced subgraph** of $G = (V, E, \alpha, \beta)$ iff (1) $W \subseteq V$ and (2) $F = E \cap (W \times W)$.

DEFINITION 2.4. An **edge set** is a labelled graph with no isolated vertex.

DEFINITION 2.5. Given $G' = (V', E', \alpha', \beta')$ and $G = (V, E, \alpha, \beta)$, a **subgraph isomorphism** is an injective function $f : (V') \rightarrow V$ such that (1) $\forall e' = (u, v) \in E'$, there exists $e = (f(u), f(v)) \in E$, (2) $\forall u \in V'$, $\alpha'(u) = \alpha(f(u))$ and (3) $\forall e \in E'$, $\beta'(e') = \beta(f(e'))$.

If there exists such a function, then G' is *subgraph isomorphic* to G . If $f : (V') \rightarrow V$ is bijective, G' is *isomorphic* to G .

DEFINITION 2.6. Given the graphs $C \subseteq G_p$, G_p and G_n , C is a **common subgraph** iff it is subgraph isomorphic to G_n .

DEFINITION 2.7. C is a **common edge set** iff (1) it is an edge set and (2) a common subgraph. C is a **maximal common edge set** iff it is a common edge set and there does not exist any strict superset which is a common edge set. C is a **maximum common edge set** if it is a common edge set and no other common edge set has more edges than C .

Note that a maximum common edge set must be a maximal common edge set, but not vice versa. Our definition of maximum common edge set is the same as the use of maximum common subgraph in the literature [9, 11].

The following definitions are presented for the case where there is a single negative graph G_n , but can be straightforwardly extended to the situation where there is a set of negative graphs $\{G_{n_1}, \dots, G_{n_k}\}$.

DEFINITION 2.8. $C \subseteq G_p$ is a **contrast subgraph** iff C is not subgraph isomorphic to G_n . It is **minimal** if all of its strict subgraphs are not contrast subgraphs.

DEFINITION 2.9. C is a **contrast edge set** iff (1) it is an edge set and (2) a contrast subgraph. It is **minimal** if all proper subsets of C do not form contrast subgraphs.

Observe that any supergraph of a minimal contrast subgraph will be a contrast subgraph.

DEFINITION 2.10. Given a graph $G = (V, E, \alpha, \beta)$, a **partition** π is a set of disjoint and non-empty subsets of V called *cells*, such that the union of all the cells in the set is V . Vertices in the same cell have the same label and vertices in different cells have different labels.

DEFINITION 2.11. Given G_p and G_n that are associated with the partitions $\pi_p = \{cell_{p_1} \dots cell_{p_j}\}$ and $\pi_n = \{cell_{n_1} \dots cell_{n_j}\}$ respectively, a **minimal contrast vertex set** is a subset of a cell in π_p such that its cardinality is exactly $|cell_{n_j}| + 1$.

A minimal contrast vertex set is a smallest possible subgraph of G_p that i) has no edges and ii) does not have any subgraph isomorphism to G_n , iii) contains vertices all with the same label. Enumerating the collection of minimal contrast vertex sets is a trivial task. The next theorem establishes a relationship between minimal contrast edge sets, minimal contrast vertex sets and minimal contrast subgraphs.

THEOREM 2.1. Given a positive graph G_p and a negative graph G_n , let $MinES$ be the set of all minimal contrast edge sets, let $MinVS$ be the set of all minimal contrast vertex sets and let $MinSG$ be the set of all minimal contrast subgraphs. Then $MinSG = (MinES \cup_{min} MinVS)$. Here \cup_{min} indicates minimal union, i.e. removal of any graphs which are supergraphs of others in the set.

3 Related Work

We will briefly review areas that are relevant to the concept of minimal contrast subgraphs.

Contrast Patterns: Contrasts have been used in other areas of data mining, such as emerging patterns, introduced by [3], which can be used to build high accuracy classification models. It was shown that there is a close relationship between the problem of computing minimal emerging patterns and minimal hypergraph transversals in [1]. We will demonstrate a similar relationship but in a graph context, using the algorithm from [1] to find the transversals as a subroutine.

Frequent Subgraph Mining: In [5], Inokuchi introduced AGM to mine all the frequent induced subgraphs. FSG, gSpan and MoFa were introduced to mine all frequent connected subgraph [8, 15, 2]. It has been shown that frequent subgraph mining can be modified to mine discriminative substructure [2]. However, the difference is these approaches are only capable of mining a subset of the contrast subgraphs (i.e. the non-isomorphic connected ones).

Maximum Common Subgraph: There are two main classes of algorithms that compute maximum common subgraphs. The first is based on the reduction of the maximum common subgraph problem to the maximum clique detection problem [4]. Another class is based on the backtracking tree, introduced by McGregor [9]. Our algorithm for finding maximal common edge sets is similar to work in [9] in its use of a backtracking

tree, but differs since it is concerned with enumerating all the maximal common edge sets, as opposed to the maximum common edge sets. Its use of ordering heuristics such as ordering of negative graphs and label ordering also differs.

Other graph contrasting approaches: Work in [12] examines how change in a time series of graphs can be determined using global distance measures. Contrasts between graphs are a focus of work in [6]. They propose a language that is able to query for fragments that are frequent in one class of graphs, but infrequent in another. A key difference from our work is that a fragment is a linear substructure of a compound. The advantage of using fragments is that the complexity of mining is lower, but the disadvantage is that richer and more complex contrasts can be missed.

4 Minimal Contrast Subgraph Miner

Our algorithm operates in three main stages. In the first stage, it discovers the maximal common edge sets between G_p and each given negative graph G_{n_i} , using a backtracking tree. Next, the maximal common edge sets across for all G_{n_i} are then unioned together and the minimal transversals of their complements computed, to yield the minimal contrast edge sets for G_p , with respect to a set of negative graphs. Finally, these contrasts are then minimally unioned with the minimal contrast vertex sets to give the complete set of minimal contrast subgraphs.

4.1 Duality Between Minimal Contrast and Maximal Common Edge Sets

We now formalise the connection between the minimal contrast edge sets and the maximal common edge sets. Suppose $S = \{S_1, S_2, \dots\}$ is a set of sets, then \bar{S} , the complement of S , is the set of complements of each of the sets. i.e. $\bar{S} = \{\bar{S}_1, \bar{S}_2, \dots\}$.

DEFINITION 4.1. Let $\mathbf{A} = \{A_1, A_2, \dots, A_n\}$ be a set of sets. We say a set P is a transversal of \mathbf{A} if $(P \cap A_1 \neq \emptyset) \wedge (P \cap A_2 \neq \emptyset) \wedge \dots \wedge (P \cap A_n \neq \emptyset)$. We say that P is a minimal transversal of \mathbf{A} , if P is a transversal of \mathbf{A} and each $P' \subset P$ is not a transversal of \mathbf{A} . We define $Min_Trans(\mathbf{A})$ to be the set of all the minimal transversals of \mathbf{A} .

The key idea is that the maximal common edge sets and the minimal contrast edge sets are duals of one another. Given a set of maximal common edge sets S , then the minimal transversals of S is equal to the collection of the smallest edge sets which are not contained in any edge sets from S (i.e. the minimal contrast edge sets). This is formalised in the following theorem.

THEOREM 4.1. Given G_p and $\{G_{n_1}, \dots, G_{n_k}\}$, let M_i be the set of maximal common edge sets between G_p and G_{n_i} . Then $Min_Trans(\bar{M}_1 \cup \bar{M}_2 \dots \cup \bar{M}_k)$ is the set of all minimal contrast edge sets between G_p and $\{G_{n_1}, \dots, G_{n_k}\}$.

4.2 Finding the Maximal Common Edge Sets

The problem reduces to finding the set of maximal common edge sets between the positive graph G_p and a negative graph G_{n_i} . We do this by using a recursive backtracking tree, similar to [9]. Each node of the tree corresponds to a tentative mapping between a vertex in the positive graph $G_p = \{V_p, E_p, \alpha_p, \beta_p\}$ and negative graph $G_{n_i} = \{V_n, E_n, \alpha_n, \beta_n\}$, as well as corresponding to a tentative common edge set. Associated with each node, is an edge correspondence matrix where each cell indicates whether an edge in G_p can correspond to an edge in G_{n_i} .

Without a priori knowledge, all cells in the matrix are set to true at the root node, indicating that all the edges can correspond to each other. Possibilities can then be eliminated using a connectivity constraint. If node i is paired with node j , all the edges connected to i can only correspond to all the edges that connected to j . A similar rule applies to edges that are not connected to i and j . At the leaf node, all the rows that do not consist of all zeros will represent an edge in the common edge set C . This algorithm is unrealistic in its basic form, because the number of possible paths is equal to $\frac{|V_n|!}{(|V_n| - |V_p|)!}$. To obtain better efficiency, more powerful pruning is needed.

Non Maximal Common Edge Set Pruning: All common edge sets must be maximal and so we should only explore paths that lead to a maximal common edge set. We therefore explore the tree in a depth first manner. Suppose we have found a common edge set C_1 and at any node of the tree, a tentative common edge set C_2 is computed. We should not proceed downwards if $C_2 \subseteq C_1$.

Ancestors and Leaf Equivalence Pruning: If the tentative common edge set of a node's parent is equivalent to the common edge set of a leaf, then there is no need to expand all the siblings of the node.

Local Maximal Common Edge Set Pruning: If there is more than one negative graph, we find the the maximal common edge sets between the positive graph and each negative graph and then keep only the maximal common edge sets across all negatives. i.e. A common edge set may be a maximal common edge set between positive graph G_p and negative graph G_{n_i} , but this may be a superset of a maximal common edge set between G_p and negative graph G_{n_j} . If G_p and G_{n_i} are compared first, we can use the maximal common edge

set found to prune unfruitful paths when later comparing G_p and G_{n_j} .

Vertex and Edge Label Pruning: If a vertex has a label that does not appear in the other graph, the edges attached to it must not be in any common edge set between the two graphs. This also applies to edges that have a distinct label. We can therefore eliminate a row or column that represents the edge in the edge correspondence matrix. Also, edges can only be paired with each other if their vertex labels at either end match. Such incompatible edge pairings are eliminated statically before tree expansion. It is also possible to eliminate distinct vertices from the mapping. Furthermore, we should only pair vertices with same label together.

Ordering Strategies: It is obvious that the earlier we reach the maximal common edge sets in the tree, the earlier pruning can be performed. The first ordering strategy considers the order in which both positive and negative vertices should be considered within the tree. For a given label, both positive vertices and negative vertices are ordered from highest to lowest degree. The intuition is that subgraphs containing vertices of high degree will likely yield larger maximal common edge sets and finding these earlier is advantageous.

The second strategy considers the order to be used across labels. Our strategy is to first try the labels which have largest distribution imbalance between the positive and negative graph. The reason is that after exhausting all the vertices with the same label in one graph, the residual vertices in the other graph that has the label cannot be mapped. Hence, they can be eliminated from mapping. In most cases, eliminating more vertices implies more edges will be eliminated from the edge correspondence matrix and so we can perform the pruning earlier in the recursion tree.

The third strategy considers the order in which the negative graphs $\{G_{n_1}, \dots, G_{n_k}\}$ should be contrasted with the positive graph G_p . We use the heuristic that larger negative graphs (having more vertices) should be compared earlier, since a larger negative graph is likely to have greater similarities with G_p and thus have larger maximal common edge sets. These edge sets can then be used to prune away common edge sets found in subsequent smaller negative graphs, using the local common edge set pruning technique.

4.3 Minimal Transversal Computation and Result Merging Once the maximal common edge sets have been computed, we need to derive the minimal contrast edge sets via the relationship described in theorem 4.1. We use the hypergraph transversal algorithm described in [1]. Once the minimal contrast edge sets have been computed, the final step is to merge them

with the minimal contrast vertex sets. The merge operation must remove any minimal contrast edge set which is a supergraph of a minimal contrast vertex set. The superset testing is very efficient, since the minimal contrast vertex sets are graphs without edges. The minimal contrast vertex sets themselves are trivial to compute, directly from definition 2.11.

5 Performance Study

We conducted a number of experiments on both real world and synthetic datasets. All experiments were run on a 2.8GHz Intel Xeon PC, with 4 gigabytes of main memory, running UNIX.

Synthetic Datasets: To study the performance of the system in a controlled environment, we generated a number of synthetic datasets with the data generator used in [8, 15] and kindly provided by Michihiro Kuramochi. We varied two principal parameters i) average transaction size (number of graph edges) and ii) number of vertex labels. The number of edge labels was fixed at 10, the number of positive graphs was fixed at 100 and the number of negative graphs was also fixed at 100. The number of vertex labels is expressed as a percentage of the average graph size. Graph 1 in Figure 1 shows the average running time when a positive graph is contrasted against 100 negative graphs (this result has been averaged over 100 different positive graphs). The running time slowly increases as graph size increases. However, the most dominant factor is the percentage of different vertex labels available.

Real World Dataset: The real world data is a subset of the dataset that was used in the Predictive Toxicology Evaluation Challenge [13]. For the description of the dataset, refer to [15, 8]. The positive class corresponds to the 65 smallest non-carcinogenic compounds and the negative class corresponds to the 65 smallest carcinogenic compounds.

Graph 2 in figure 1 shows the runtime when the n smallest positive graphs are contrasted against the smallest n negative graphs. This time is divided by the number of positive graphs in each case (so each data point effectively represents $1 \times n$ graph comparisons, where n is the value of the x-axis). We can see that as the positive graph becomes larger, the running time increases exponentially. The running time is negligible for the very small graphs.

6 Discussion

Comparing the running time results for this real world data with those obtained for the synthetic data, a clear difference is apparent. The former is far much more demanding. Indeed all contrasts for synthetic graphs of

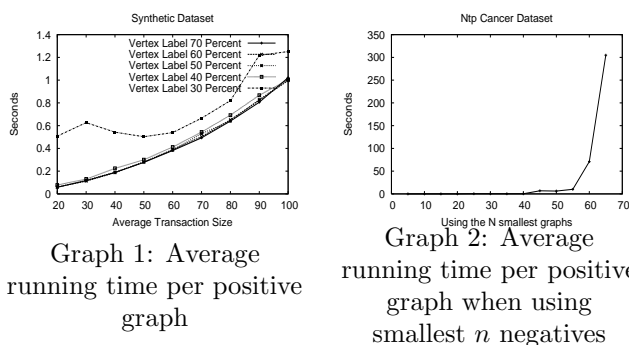


Figure 1: Experiment results

size 100 can be quickly mined, but real world graphs of size more than 20 (the 65th graph) pose a significant challenge. The reason is that most graphs in the synthetic dataset contain vertices of all available label types. However, most graphs in the chemical compound dataset only contain a very small percentage of the total number of available labels. This causes the the process of finding maximal common edge sets in the chemical compound dataset to be much harder, since for a given positive/negative graph pair, the use of vertex label pruning is much less effective for pruning search space within the backtracking tree. We have also performed a number of other experiments, not described here due to lack of space. These are described in the full version of this paper.

One technique for reducing the mining complexity is to mine contrast graphs that satisfy a minimum support threshold in the positive set of graphs. Such contrasts could be easily found via an additional preprocessing step, whereby an existing tool such as [15] or [8] is used to find frequent subgraphs satisfying a minimum support constraint in the positive set of graphs. Each of the resulting graphs is then individually contrasted against the entire negative set, using *ConSubGraphMiner*. Any minimal contrast subgraphs then found would be guaranteed to satisfy the minimum positive support constraint, as well as the zero negative support constraint. However, the solution may not be complete because the preprocessing tools can only mine frequent connected and non-isomorphic subgraph.

7 Conclusions

We have introduced the data mining problem of *minimal contrast subgraphs*. These patterns capture essential contrast information between collections of graphs. We have formally established the relationships between minimal contrast subgraphs and minimal contrast edge sets and also showed a duality that exists between minimal contrast edge sets and maximal common edge

sets. We presented an algorithm for mining minimal contrast subgraphs and demonstrated its ability to mine contrasts from both real world and synthetic data.

Acknowledgements: This work is partially supported by National ICT Australia. National ICT Australia is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

References

- [1] J. Bailey, T. Manoukian, and K. Ramamohanarao. A fast algorithm for computing hypergraph transversals and its application in mining emerging patterns. In *ICDM 2003*, pages 485–488.
- [2] C. Borgelt and M. R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *Proc. of ICDM 2002*, pages 51–58.
- [3] G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *KDD 1999*, pages 43–52.
- [4] P.J. Durand, R.Pasari, J.W.Baker, and Chun che Tsai. An efficient algorithm for similarity analysis of molecules. *Internet Journal of Chemistry*, 2, 1999.
- [5] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *PKDD 2000*, pages 13–23.
- [6] S. Kramer, L. De Raedt, and C. Helma. Molecular feature mining in HIV data. In *KDD 2001*, pages 136–143.
- [7] Evgeny B. Krissinel and Kim Henrick. Common subgraph isomorphism detection by backtracking search. *Softw., Pract. Exper.*, 34(6):591–607, 2004.
- [8] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *ICDM 2001*, pages 313–320.
- [9] J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Soft. Practice and Experience*, 12(1):23–34, 1982.
- [10] B. D. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.
- [11] J. W. Raymond and P. Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *J. of Computer-Aided Molecular Design*, 16(7):521–533, 2002.
- [12] P. Shoubridge, M. Kraetzl, W. Wallis, and H. Bunke. Detection of abnormal change in a time series of graphs. *J. of Interconnection Networks*, 3(1–2):85–101, 2002.
- [13] A. Srinivasan, R. King, S. Muggleton, and M. Sternberg. The predictive toxicology evaluation challenge. In *IJCAI 1997*, pages 4–9.
- [14] Julian R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, 1976.
- [15] X. Yan and J. Han. gSpan: Graph-based substructure pattern mining. In *ICDM 2002*, pages 721–724.