UNIVERSITY OF CALIFORNIA

Los Angeles

Active Learning for the Subgraph Matching Problem

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Mathematics

by

Yurun Ge

2023

ABSTRACT OF THE DISSERTATION

Active Learning for the Subgraph Matching Problem

by

Yurun Ge
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2023
Professor Andrea Bertozzi, Chair

The subgraph matching problem arises in a variety of domains including pattern recognition for segmented images, meshes of 3D objects, biochemical reactions, and security applications. Large and complex solution spaces are common for this graph-based problem, especially when the world graph contains many more nodes and edges in comparison to the template graph.

Overall, this dissertation covers three parts: using symmetry to boost the subgraph matching algorithm and compress the solution space; presenting an active learning problem for subgraph matching problem; and discussing the different quantitative strategies for active learning.

As for the symmetry, we introduce rigorous definitions of structural equivalence and establish conditions for when it can be safely used to generate more solutions. We then adapt a state-of-the-art solver and perform a comprehensive series of tests to demonstrate how the Venn-diagram could be applied to visualize the symmetry and compress the solution space.

Next, a real use-case scenario may require analysts to query additional information to reduce the complexity of the problem and there is currently little guidance to analysts on how to optimize this choice. By analogy to the well-known active learning problem in machine learning classification problems, we present an active learning problem for the subgraph matching problem in which an algorithm suggests optimal template target nodes that would be most likely to reduce the overly large solution space. Additional information about those target nodes are then acquired from humans in the loop, in an iterative process. We present some case studies illustrating different strategies on a variety of datasets.

We further develop this idea by introducing rigorous mathematical definitions of the active learning problem. We also prove NP completeness of this problem. We present numerical experiments for single channel and multichannel subgraph matching problems created from both synthetic and real world datasets. We compare different quantitative criteria for choosing nodes to query. We introduce a new method based on a spanning tree that outperforms other graph-based criteria for the multichannel datasets.

The dissertation of Yurun Ge is approved.

Wei Wang

Deanna M. Hunter

Marcus Leigh Roper

Andrea Bertozzi, Committee Chair

University of California, Los Angeles

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

We thank Kevin Miller for helpful conversations regarding active learning for statistical machine learning. This material is based upon work supported by the US National Science Foundation under Grants No. DMS-2027277 and DMS-2318817. The sponsor had no involvement in the study design; in the collection, analysis and interpretation of data; in the writing of the report; and in the decision to submit the article for publication. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the US National Science Foundation.

VITA

2014-2018 B.S. in Mathematics in Shanghai Jiaotong University, China

PUBLICATIONS

Applications of Structural Equivalence to Subgraph Isomorphism on Multichannel Multigraphs T. Nguyen, D. Yang, Y. Ge, H. Li and A. L. Bertozzi, 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 2019, pp. 4913-4920, doi: 10.1109/BigData47090.2019.9006538.

Active Learning for the Subgraph Matching Problem Y. Ge and A. L. Bertozzi, "," 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 2021, pp. 2641-2649, doi: 10.1109/BigData52589.2021.9671760.

Structural Equivalence in Subgraph Matching D. Yang, Y. Ge, T. Nguyen, D. Molitor, J. D. Moorman and A. L. Bertozzi, in IEEE Transactions on Network Science and Engineering, vol. 10, no. 4, pp. 1846-1862, 1 July-Aug. 2023, doi: 10.1109/TNSE.2023.3236028.

CHAPTER 1

Introduction to Subgraph Matching

1.1 Background of the subgraph matching problem

The subgraph matching problem, also known as the subgraph isomorphism problem is an important problem in computer science and graph theory as illustrated in the surveys [24],[14],[23]. It has practical applications in many areas such as pattern recognition [68], image analysis [85], bioinformatics [22], and knowledge graph searches [1, 78]. In the subgraph matching problem, we consider two networks: a template graph and a world graph. We wish to find all subgraphs of the world graph that match the template. To represent the broad variety of graphs in the real world, we allow the nodes in the graphs to be labelled as well as permit any number of edges in either direction between nodes.

In this dissertation, we consider the subgraph matching problem for multichannel graphs, meaning that each edge in a graph occupies a specific "channel" that represents a type of edge, e.g. email communication versus electronic fund transfer. Multichannel graphs are used in adversarial activity detection[59] with complex scenarios in which edge types represent different types of activities. Another type of graph that has different edge types are knowledge graphs in which edges can represent relationships between entities (nodes). Such graphs also can have an ontology that describes a heirarchical structure of labels on the graph. Figure 1.1 depicts an simple example of the subgraph isomorphism problem for multichannel graphs.

Multichannel graphs are closely related to single channel graphs with edge labels. For a single channel graph, we can construct a multichannel graph by putting together the edges with the same labels in the same channel, and vise versa. These two notions are different in the following sense: (1) edge labels may possibly exist in the multichannel graphs, and for these graphs with edge labels, they cannot be easily converted to single channel graphs. (2) the topology structures of the multichannel graphs are different from single channel graphs.

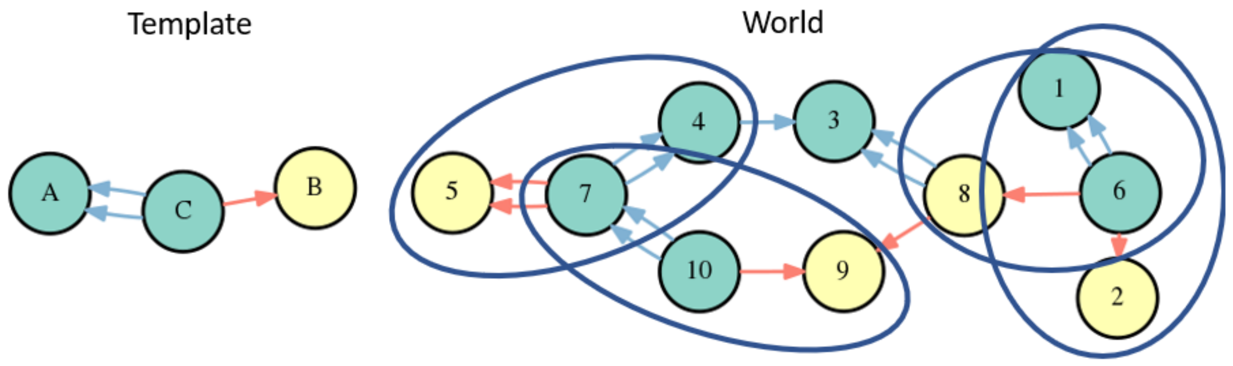

Figure 1.1: Example subgraph matching problem in the multichannel setting reproduced from [59].(© 2021 IEEE) The colors of nodes and edges correspond to different node labels and channels, respectively. There are four subgraph isomorphisms corresponding to mapping the template nodes (A, C, B) to each of the four circled sets of nodes $(4, 7, 5), (7, 10, 9), (1, 6, 8), (1, 6, 2)$.

Part of this research was supported by the DARPA Modeling Adversial Activity (MAA) program [62]. The MAA program desires to develop graph models and algorithms in order to better understand and detect transactions which can potentially be associated with the development of "weapons of mass terror". Some of the data sets referenced in this thesis are generated by three performers for that program. the PNNL (Pacific Northwest National Laboratory [17] , the GORDIAN (Graphing Observables from Realistic Distributions in Activity Networks) project [42], and Ivysys Technologies [2]). These teams generate template graphs and world graphs from both real life and synthetic data like COVID data [93], and detecting the adversarial activity is the same as solving the subgraph matching problem. However, these datasets in practice often possess high computational complexity, making them unsolvable with standard subgraph isomorphism approaches. In addition to the DARPA datasets, we also consider multichannel public domain datasets such as the British Transportation Network[26], Higgs Twitter network [21], Airlines [9].

1.2 Solving the Subgraph matching problem

In the first section, we introduce the definition of a multichannel graph and the subgraph matching problem for multichannel graph. This section subsection provides the formal mathematical definition of the subgraph matching problem and reviews the standard algorithms from the previous research for solving the subgraph matching problem.

Definition 1 (Multichannel Graph)**.** *A **multichannel graph** $G = (V, E, L, C)$ is a set of nodes (or vertices), directed edges between the nodes, labels on the nodes, and channels on the edges. The number of nodes is denoted n. Each node $\mathbf{v} \in V$ has a label $L(\mathbf{v})$ belonging to some arbitrary set of labels. There can be any number of edges between each pair of nodes $(\mathbf{u}, \mathbf{v})$ in either direction. Each edge belongs to one of the channels C. Edges between the same pair of nodes in the same channel with the same direction are indistinguishable. The function $E : V \times V \to \mathbb{N}^{|C|}$ describes the number of edges in each channel between each pair of nodes. In particular, $E(\mathbf{u}, \mathbf{v})$ can be represented as a $|C|$-dimensional vector the k^{th} element of which is the number of edges from node $\mathbf{u}$ to node $\mathbf{v}$ in the k^{th} channel.*

We refer to graphs with only one channel as **single-channel graphs**.

Now we introduce some standard graph terminology which we extend to the multichannel setting. Two vertices are **adjacent** or **neighboring** if there is an edge between them in either direction in any channel. An edge is **incident** to a vertex if the vertex is one of the edge's endpoints. We define the **degree** of a vertex v, denoted $\deg(v)$, in a multichannel graph as the number of edges incident to a vertex. The neighborhood of a vertex v, denoted $N(v)$, is the set of all adjacent vertices. A **path** is a finite sequence of distinct vertices such that any two consecutive vertices are adjacent. A **subgraph** of a multichannel graph $G = (V, E, L, C)$

is a multichannel graph $G' = (V', E', L', C')$ with $V' \subset V, C' = C$, $L'(v) = L(v)$ for all $v \in V$, and $E'(u, v) \leq E(u, v)$ for all $u, v \in V'$.

The **subgraph matching problem** (SMP) can be succinctly stated: given two multichannel networks, a template $G_t = (V_t, E_t, L_t, C)$ and a world $G_w = (V_w, E_w, L_w, C)$, we wish to find all subgraphs of the world that *match* the template. To formalize what we mean by match, we introduce the *subgraph isomorphism (SI)* as defined in [59].

Definition 2 (Subgraph Isomorphism)**.** *An injective function $f : V_t \to V_w$ is called a **subgraph isomorphism (SI)** or **subgraph matching** from $G_t = (V_t, E_t, L_t, C)$ to $G_w = (V_w, E_w, L_w, C)$ if*

$$L_t(\mathbf{v}) = L_w(f(\mathbf{v})) \qquad \forall \mathbf{v} \in V_t \tag{1.1}$$

$$E_t(\mathbf{u}, \mathbf{v}) \leq E_w\left(f(\mathbf{u}), f(\mathbf{v})\right) \qquad \forall \mathbf{u}, \mathbf{v} \in V_t \times V_t. \tag{1.2}$$

The set of all SIs from G_t to G_w is denoted $F(G_t, G_w)$.

This definition allows for isomorphisms in which the world graph has more edges than the template. If the function additionally satisfies (1.2) at equality, it is an **induced subgraph isomorphism**. A function which satisfies (1.1) and (1.2) but is not necessarily injective is an **edge preserving map (EPM)**.

Finding a subgraph matching is proved to be NP-complete [28]. As a result, there is no algorithm that efficiently finds all subgraph matchings on all graphs. Standard algorithms for exact subgraph matching use one of three approaches [10, 11]: tree search, constraint propagation, and graph indexing [37].

Tree search methods was first applied to solving the subgraph matching problem in single channel undirected graph in Ullmann's algorithm [79]. For each template node, a tree search algorithm creates a list of nodes in the world graph that enumerates all the possible mapping of the template node. This list is called candidate list. The tree search algorithm then exhausts all the possible mappings of the template node. In each step of the tree search, the template node is mapped to one of the nodes in its candidate list, and then all the remaining candidate lists are refined. The tree search algorithms have different efficiency due to their different ways to create and refine candidate list. For a given template node, the Ullman's algorithm includes all the nodes in the world graph with degree no less than the template node. In each step of the tree search, the candidate lists are refined by removing the candidates that are impossible to be neighbors for a neighboring pair in the template graph. This algorithm is further developed in , VF2 [16] and its variants (VF2Plus [12], VF3 [10, 11], VF2++ [40]). In VF2 and its generalization methods, the algorithms can be used to solve subgraph matching problem in directed graphs. Also, more semantic rules are applied to refine the candidate sets, like the rules considering the matched nodes (nodes with only one candidate) and the rules that counts the number of valid neighbors of a candidate.

As for the tree search of the candidate list, the order to select nodes has a huge impact on the algorithm efficiency. The approach to determine the best index of the template graph nodes in the tree search to speed up algorithms is referred to as the graph indexing approach. Examples of graph indexing approaches include GraphQL [34], SPath [91], GADDI [89], QuickSI [72], TurboISO [32], BoostISO [66], CFL-Match [6], and CNI-Match [60]. These methods exploits the graph structures and data structures in the template graph and their candidate lists. Many of them have inspired our work. In [66], the authors proposed algorithm which makes use of the equivalence class from the perspective of set theory. They defined syntactic equivalence to combine the similar nodes in the template graph structure and the query dependent equivalence based on the graph structure and the candidate list. They also proposed the syntactic containment and query containment relations. These relations reduce the redundant computation in the tree search. Also, in CFL-Match [6], the authors found that it improves the computation efficiency to decompose the template graph into three parts: core, forest and leaves and follow a specific search order from nodes in core to forest then leaves. It is worth mentioning that in their work, they proposed an innovative data structure called CPI (compact path index) to encode possible matchings of the template graph. In the CPI structure, an auxiliary graph is constructed among the candidates of all the nodes in the template graph, where the adjacent pair of candidates of the adjacent template graph are connected. In their work, the CPI structure is constructed on a breadth first search (BFS) tree of the template graph by visiting the nodes level-by-level in a top-down manner. This structure plays an important role in determining the order of the three search and computing the number of subgraph matchings. In [87], the authors defined the candidate structure for the purpose of exploring the topology structures among candidates. The candidate structure is also an auxiliary graph constructed based on adjacent template pairs and their candidates. The formal definition is given as follows:

Definition 3. *Given template graph $G_T = (V_T, E_T)$, world graph $G_W = (V_W, E_W)$, and candidate sets $C[u] \subset V_W$ for each $u \in V_T$, the **candidate structure** is the directed graph $G_C = (V_C, E_C)$ where the vertices $V_C = \{(u, c) : u \in V_T, c \in V_W\}$ are template vertex-candidate pairs and $((u_1, c_1), (u_2, c_2)) \in E_C$ if and only if $(u_1, u_2) \in E_T$ and $(c_1, c_2) \in E_W$.*

Constraint propagation approaches cast the problem as a constraint satisfaction problem. One keeps a record of world nodes that are possible matches for each template node.This approach can be combined with a tree search to solve the subgraph matching problem. Examples of constraint propagation approaches include McGregor [55], nRF+ [46], ILF [88], LAD [73] (and its variants, IncompleteLAD and PathLAD [44]), McCreesh and Prosser (Glasgow) [52], and FocusSearch [80]. These methods differ from each other in the constraint rules they apply. For example, in Larrosa's work [46], they proposed a constraint saying that for a candidate of the template node, the number of its neighbors must be less than or equal to the number of candidates for those neighbors that are adjacent to the candidate. The LAD method [44] designed a constraint that for a template node and its candidate, their should at least one solution to map their neighbors without duplicating. This constraint is equivalent to solving an all-different problem.

After checking the constraints , the algorithm uses the tree search to find all the subgraph matchings. A typical algorithm that was used in these papers is shown in 1 . In this algorithm, the "Solve" function is executed recursively after each time we "match" a template graph node u to its candidate v. The rule of applying the constraints is also referred to as "filters" in the tree search algorithm. The function "ApplyFilters" reduces the candidate list based on the constraint rules. The tree search algorithm will terminate until all the possible matchings of template nodes to its candidates are exhausted.

Algorithm 1 Basic routine for a tree search

```
1: function SOLVE(partial_match, cands)
2:     if MatchComplete(partial_match) then
3:         ReportMatch(partial_match)
4:         return
5:     ApplyFilters(partial_match, cands)
6:     Let $u$ = GetNextTemplateVertex()
7:     Let cands_copy = cands.copy()
8:     for candidate $v$ of $u$ do
9:         partial_match.match($u, v$)
10:        Solve(partial_match, cands_copy)
11:        partial_match.unmatch($u, v$)
12:        for unmatched $u' \sim u$ do
13:            Set cands[$u', v$] = 0
14:    Let cands = cands_copy
15:    return
```

Our work in [57, 59] designed and generalized constraint rules to multichannel graphs. The rules include statistic filter, topology filter, repeated sets filter, neighborhood filter and elimination filter. The statistic filter checks the satisfaction of the node-level statistics such as in/out-degree, number of in/out-neighbors, number of reciprocated edges, and number of self-edges. The topology filter checks the existence of the mapping from the candidate lists of the following structure: a template node and all its neighboring nodes. The repeated sets filter removes redundant candidates by checking the already matched template nodes or group of nodes. The neighborhood filter checks the existence of injective mapping from the neighbors of a template node to their candidates under which the corresponding nodes in the world graph are also neighbors. It is also called the all-different constraint. The elimination filter will assign a template node to its candidate node and remove the candidate node if no mapping would exist after running other filters.

The following figure 1.2 shows how the number of candidates change after applying the different types of filters. The datasets used in this experiment is from [17].

In practice, both real world and synthetic examples illustrate the complexity of the set of solutions to the subgraph matching problem. For example, for single channel (single edge-type) networks, there are several benchmark datasets [74, 18, 75, 29, 53] for which the total SI count ranges from zero to approximately 10^{384} [87]. This complexity arises from two key features of the problem. On the one hand, the world graph may have additional nodes and edges that are equally good candidates for components of the template graph. On the other hand the template has nodes and edges that are interchangeable. These are forms of equivalence that have been explored recently in the literature [78, 87] to reduce the complexity of the solution space by providing a categorization of groups of nodes that can be interchanged. In figure 1.3 is an example of the template graph that consists of interchangeable nodes. In our work [87], we show that by considering the interchangeability among the equivalence classes in this dataset, one can generate over 10^{100} subgraph matchings from a single representative solution.

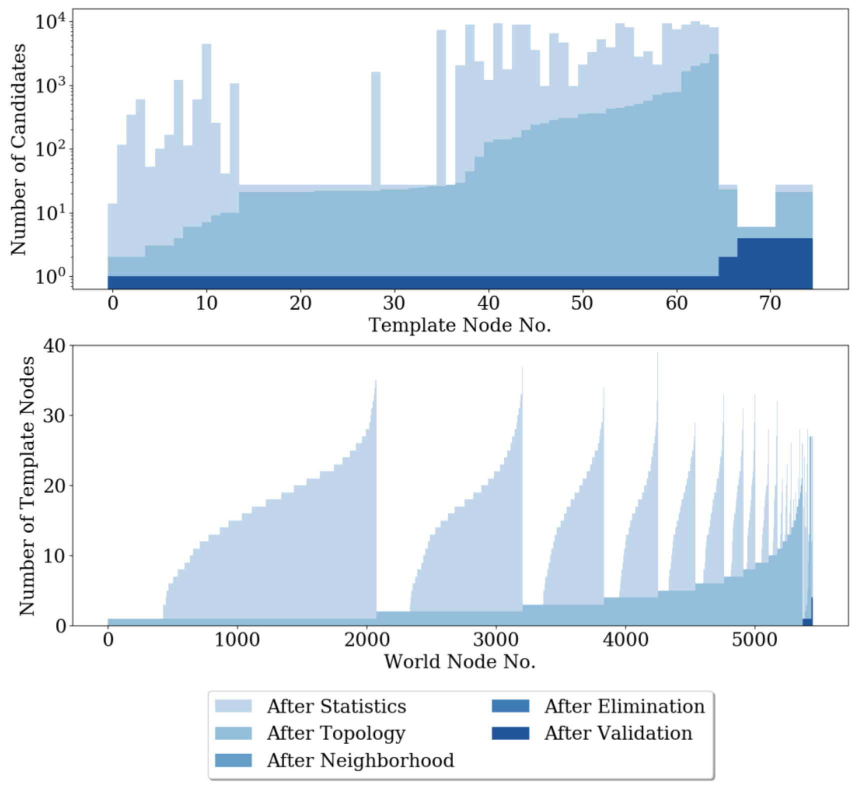

Figure 1.2: (Top): The number of candidates for each template node after different levels of filtering are applied to PNNL Version 6 B1-S1. (Bottom): The number of template nodes for which each world node is a candidate.. Note that the Validation histogram perfectly overlaps the Elimination histogram and the Neighborhood histogram perfectly overlaps the Topology histogram. This figure is from [59] (© 2021 IEEE)

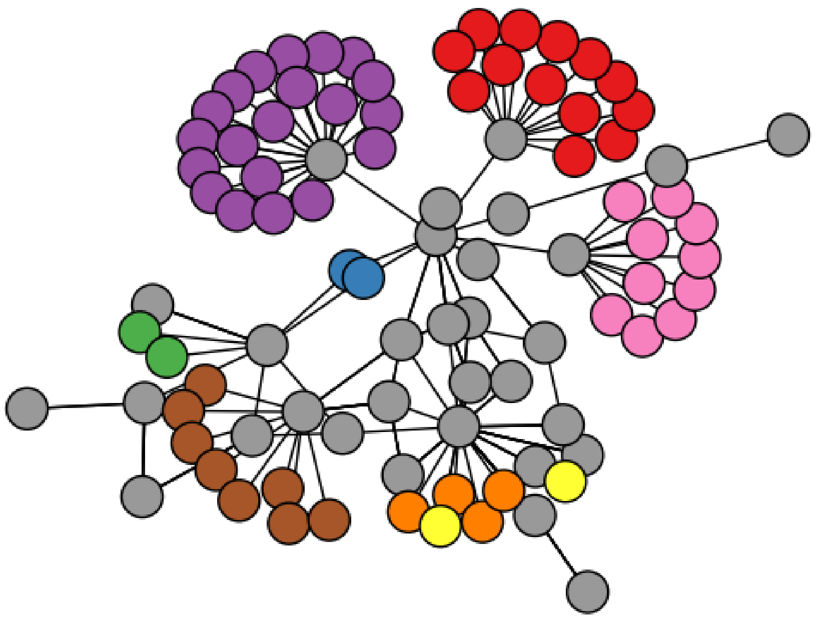

Figure 1.3: Example of equivalence class from [87](© 2023 IEEE). The template graph is from Ivysys Technologies [2]. Non-gray vertices of the same color represent different equivalence classes and are interchangeable without changing the structure of the graph.

1.3 The current research landscape of combinatorial graph problems

1.3.1 Related Combinatorial Graph Problems

While this dissertation focuses on exact matches for subgraph matching, there is a literature on the relevant problems. The work discussed in this thesis could be possibly extended to these similar problems.

One closely related problem is the inexact subgraph matching problem. This problem replaces the strict edge-preserving constraints of exact isomorphism with a penalty for edge and label mismatches which is to be minimized. The exact form of the penalty varies across applications and there are a myriad of approaches many taking inspiration from the exact matching problem. These involve a variety of techniques including filtering [78, 43], A* search [39], indexing [47], and continuous techniques [76].

Another related problem is the maximum common subgraph detection problem. This problem deals with two input graphs and aims to find their largest common subgraphs. In this literature [3], the authors designed a "branch and bound" algorithm called GLSearch based on Graph Neural Network. Their work shows that combining search methods with reinforcement learning and deep learning improves the efficiency and is promising in solving these constraint combinatorial problems.

In a recent work [45], the authors developed efficient learning methods called AEDNet for obtaining a sufficient similar solution for the induced subgraph matching problem. An induced subgraph is a special case of the subgraph matching in which the edge count in the template and world are the same for those nodes in the template and its image. They justify the need for an efficient solver for subgraph matching problem in large graphs to satisfy the need for real life use, and their method is much faster than the exact solver in large graphs.

1.3.2 Variability in Scale of Datasets for Subgraph Matching

The dataset has a significant influence on the performance of subgraph matching algorithms. On the one hand, researchers develop different methods that are applicable to the template graph and world graph with different scales. For instance, while [45] presents template graphs that are 20 times larger than previous methods, these are still much smaller than those in our study, making their approach less suited to our datasets. On the other hand, even for the template graphs and world graphs with similar sizes, the level of difficulty to solve the subgraph matching problem can vary significantly. For example, the work of [59] shows that for different instances in the PNNL[17] dataset with similar template graph size and world graph size, the number of total isomorphisms can range from 1152 to 3.13×10^8.

In the following paragraphs, we enumerate the scale of datasets used in some notable state-of-art researches on subgraph matching.

Liu and his team proposed a learning based method to count the total number of all the subgraph matchings. They significantly accelerated the algorithm of VF2 [16] with the help of their learning framework. Their work is tested on some "small" datasets and some "large" datasets. For their"small" datasets, the world graph has an average of 76.3 vertices. The total number of matchings is less than 1024. The corresponding statistics for the world graph in their "large" datasets is 560, and the total number of matchings is less than 4096. The template graphs of the subgraph matching problem have sizes from 4 to 16. They are generated using randomized algorithms by the research group.

As introduced in the above subsection, the AEDNet algorithm [45] tries to find sufficiently similar matchings of an induced subgraph. This problem is considered to be less complex than the general subgraph matching problem for the following reasons. First, this problem allows some errors on the matchings they find. Second, the requirement of the induced subgraph is a strong constraint on the problem that would greatly trim the tree search. Their algorithm is tested across six datasets with the average number of vertices in the template graph ranging from 7.5 to 75 and in the world graph from 19.77 to 2372.

The work [90] also studies learning based methods for counting all the number of subgraph matchings. They combine their algorithm with active learning in order to accelerate the training of deep neural network. Their algorithm is tested on datasets with larger world graphs. The largest world graph in their datasets contains 12,811,197 vertices, 15,768,516 edges and 188,883 different labels. However, the sizes of their template graph range from 3 to 12, which are small compared to datasets in other research and in this thesis.

In table 1.1, we list the statistics of datasets in the works mentioned earlier in this section. These datasets are the largest in the cited paper using the metric of template graph size, world graph size, or the number of matchings. We also list two large datasets BTN100 and Ivysys v7 used in this thesis. The detail of these datasets are introduced in chapter 3 and 4. From the comparison, the datasets in our research have distinct characteristics compared to those in state of the art research regarding to the number of matchings and template size.

Table 1.1: Comparison of Dataset scales on Subgraph Matching. The first four datasets are from state-of-art algorithms introduced in section 1.3.2. The last two datasets, BTN100 and Ivysys are the datasets used in this thesis. The first column represents the dataset name. The second column represents the maximum number of nodes in the template graph. The third column represents the average number of nodes in the world graph. The fourth column represents the number of labels or channels. The fifth column represents the estimated number of subgraph matchings. "Unknown" indicates that the authors have not estimated the number for various reasons.

Dataset	**T Size(max)**	**W Size(avg)**	**# of channels**	**# of matchings**
"Large" in [48].	16	560	1	4,096
PPI in [45]	75	2,372	1	Unknown
yago in [90]	8	12,811,197	188,883	10^8
eu2005 in[90]	12	842,664	40	10^{14}
BTN100	100	262,377	7	Unknown
Ivysys v7	94	2,488	3	$> 10^{103}$

CHAPTER 2

Equivalence Structure in Subgraph matching

In this chapter, we introduce our work in [87]. This work studies the acceleration of subgraph matching algorithms with the different equivalence relation among the nodes in the template graph and world graph, and the compress of the subgraph matching solutions with the equivalence structure. I contributed to proposing a novel equivalence relation called the node cover equivalence, as well as the compressed representative solution using the node cover equivalence. I also developed a Venn diagram representation for the solution space, making use of structural equivalence.

2.1 Structural equivalence and candidata equivalence

In our work, we study three kind of equivalences: structrual equivalence, candidate equivalence and node cover equivalence. The idea of the equivalence structure is to explore the nodes that are interchangeable with each other in a subgraph matching solution. In this way, if we have one representative subgraph matching from the template graph to the world graph, we can generate other matchings by swapping the nodes in the equivalence class. The definitions of these relations are listed as followed:

Definition 4. *In a graph $G = (V, E)$, we say that two vertices v, w are **structurally equivalent** (denoted $v \sim_s w$) if:*

1. *For $u \in V, u \neq v, w$,*

    (a) $(u, v) \in E \Leftrightarrow (u, w) \in E$.

(b) $(v, u) \in E \Leftrightarrow (w, u) \in E$.

2. $(v, w) \in E \Leftrightarrow (w, v) \in E$.

This definition implies that the neighbors of structurally equivalent vertices (not including the vertices themselves) must coincide. Nodes in template graph and world graph that are structurally equivalent can be freely interchangeable with each other in a subgraph matching solution and thus generating factorial numbers of other solutions.

The candidate equivalence is defined based on the candidate structure3 which was introduced in the previous section.

Definition 5. *Given a candidate structure $G_C = (V_C, E_C)$, $c_1, c_2 \in V_W$, we say that c_1 is **candidate equivalent** to c_2 with respect to $u \in V_T$, denoted $c_1 \sim_{c,u} c_2$, if and only if $c_1, c_2 \notin C[u]$ or $c_1, c_2 \in C[u]$ and $(c_1, u) \sim_s (c_2, u)$.*

We can swap the candidates in the candidate structure that is specified in the following proposition:

Proposition 6. *Suppose that given a specific candidate structure $G_C = (V_C, E_C)$, we have that $c_1, c_2 \in C[u]$ and $c_1 \sim_{c,u} c_2$ for some template vertex u. Suppose that c_1 and c_2 are not candidates for any other vertex. Then c_1 and c_2 are G_C-interchangeable.*

Proof. This is clearly injective, and for any edge (v, w) which doesn't include u, g agrees with f, so it preserves those edges. If we have $(u, v) \in E_T$, we have $(f(u), f(v)) = (c_1, f(v)) \in E_W$. Since $c_1 \sim_{c,u} c_2$, and we have $((u, c_1), (v, f(v)) \in E_C$, we must have $((u, c_2), (v, f(v)) \in E_C$ which implies $(c_2, f(v)) \in E_W$. $\square$

In the above proposition, we add one additional condition that c_1 and c_2 are not candidates for any other vertex. Without the condition, the interchangeability situation is too complex to discuss. So, the above candidate equivalence is dependent on the tree search.

In our tree search, when we generate candidate vertices for a given vertex u, we find representatives, for each candidate equivalence class, that do not appear as candidates for other vertices. If a class has a vertex appearing in other candidate sets, then we cannot exploit equivalence and must check each member of the class. Furthermore, as we continue to make matches and eliminate candidates, more world vertices will become equivalent, so it is advantageous to recompute equivalence in each step of the tree search.

If we have that $f(v) = c_1$ and $f(w) = c_2$, and we want to swap c_1 for c_2, we need a stronger condition; namely, we need that they are equivalent with respect to both v and w. In the process of a tree search, we do not know exactly what each vertex will be mapped to so instead we consider an even stronger condition:

Definition 7. *Given a candidate structure $G_C = (V_C, E_C)$, we say that $c_1 \in V_W$ is **fully candidate equivalent** to $c_2 \in V_W$, denoted $c_1 \sim_c c_2$ if for all $u \in V_T$, $c_1 \sim_{c,u} c_2$.*

Proof. If $(x, y) \in E_T$, and neither is u_1 or u_2, then g agrees with f and preserves the edge. If one is u_1 or u_2, without loss of generality take $x = u_1$, then $(g(x), g(y)) = (c_2, f(y))$. As f is a subgraph isomorphism $(c_1, f(y)) \in E_W$ and since $c_1 \sim_c c_2$, $(c_2, f(y)) \in E_W$ as well. If $x = u_1, y = u_2$, then $(g(x), g(y)) = (c_2, c_1) = (f(y), f(x))$ and this edge is in E_W since $c_1 \sim_c c_2$ and $(c_1, c_2) \in E_W$. □

Note that if $c_1 \sim_{c,u} c_2$ for some u, and c_1, c_2 are not candidates for any other vertices, then $c_1 \sim_c c_2$. This condition enables us to interchange world vertices and still maintain the subgraph matching conditions. This is established by the following proposition:

Proposition 8. *Suppose that given a specific candidate structure $G_C = (V_C, E_C)$, we have that $c_1, c_2 \in V_W$ and $c_1 \sim_c c_2$. Then, c_1 and c_2 are G_C-interchangeable.*

2.2 Node Cover Equivalence

An alternate notion of equivalence, introduced in [59], involves the use of a node cover. A **node cover** is a subset of nodes whose removal, along with incident edges, results in a completely disconnected graph. The approach in [59] is to build up a partial match of all the vertices in the node cover followed by assigning all the nodes outside the node cover. After reducing the candidate sets of all the nodes outside the cover to those that have enough connections to the nodes in the cover, what remains is to ensure that they are all different.

We formalize this with some definitions. A **partial match** is a subgraph isomorphism from a subgraph of the template graph to the world graph. We list out the mapping as a list of ordered pairs $M = \{(v_1, w_1), \dots, (v_n, w_n)\}$. A template vertex - candidate pair (v, c) is **joinable** to a partial match M if for each $(v_i, w_i) \in M$, if $(v_i, v) \in V_T$, then $(w_i, c) \in V_W$ and if $(v, v_i) \in V_T$, then $(c, w_i) \in E_T$. If two world vertices w_1, w_2 are interchangeable in any subgraph isomorphism extending a partial match M, we say that w_1 and w_2 are **M-interchangeable**.

Since the problem is significantly simpler, it is easier to obtain a form of equivalence on the vertices.

Definition 9. *Let M be a partial match M on a node cover N of V_T and suppose that for all $u \in V_T \setminus N$, the candidate set $C[u]$ is comprised entirely of all world vertices joinable to M. Two world vertices w_1, w_2 are* ***node cover equivalent*** *with respect to M, denoted $w_1 \sim_{N,M} w_2$, if for all $u \in V_T \setminus N$, $w_1 \in C[u]$ if and only if $w_2 \in C[u]$.*

For example, consider the template and world in figure 2.1. Once nodes B and D in the node cover are mapped to 2 and 5, the remaining nodes have candidates that have the associated color in the world graph. We then simply group each of these candidates together into equivalence classes. Note that the edges depicted in red are what prevent structural equivalence, and the node cover approach effectively ignores these edges to expose the equivalence of these vertices.

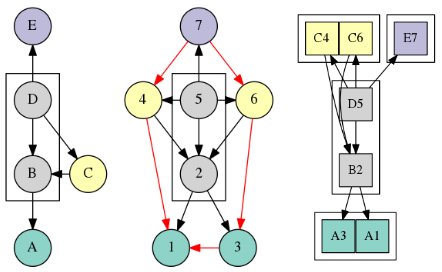

Figure 2.1: In order from left to right: template, world, and possible candidate structure. The boxed vertices comprise a node cover of the template and the image of the node cover in the world. Nodes of the same color in the world are node cover equivalent. The red edges are extraneous edges which once removed, expose equivalence.This figure is from [87](© 2023 IEEE)

Proposition 10. *Suppose we have a node cover of the template graph N, and a partial matching M on N and two world vertices w_1, w_2 not already matched satisfy $w_1 \sim_{N,M} w_2$. Then w_1 and w_2 are M-interchangeable.*

Proof. Let f be such an isomorphism which maps u_1 to w_1 and u_2 to w_2 and g interchanges w_1 and w_2. Consider $(x, y) \in E_T$. If neither are u_1, u_2, then g agrees with f and so the edge is preserved. If one of them is u_1 or u_2, say $x = u_1$, then it must be that y is in N as N is a node cover (u_1 is disconnected from any element outside the node cover). Since f is a subgraph isomorphism, (u_1, w_1) must be joinable to M and $(w_1, f(y)) \in E_T$ and so $w_1 \in C[u_1]$. It must be that $w_2 \in C[u_1]$ and so (u_1, w_2) is also joinable to M. Hence $(w_2, f(y)) = (g(x), g(y)) \in E_T$. The last case $x = u_1$ and $y = u_2$ is impossible since x and y are outside the node cover and therefore disconnected. □

Node cover equivalence is easy to check and captures a significant portion of the equivalence posed by other methods. This is often due to interchangeable nodes being composed of sibling leaves which are generally outside of a node cover.

We note that the methods discussed in this chapter (with the exception of basic structural equivalence for template and world) cannot incorporate both template and world equivalence. The combination of allowing template and world node interchanges and having dynamic world equivalence classes significantly complicates the counting process. One approach that can facilitate the use of both forms of equivalence involves a tree search where entire template equivalence classes are assigned at once instead of individual template nodes. This kind of approach for assigning the remaining nodes outside of the node cover is discussed in the Appendix Section D of [87].

2.2.1 Equivalence Hierarchy

There is a relation between node cover equivalence and full candidate equivalence, given in the following proposition:

Proposition 11. *Suppose that N is a node cover of V_T, M is a partial match on N, candidate sets are reduced to joinable vertices, and $w_1, w_2 \in V_T \setminus N$. Then $w_1 \sim_{N,M} w_2 \Leftrightarrow w_1 \sim_c w_2$.*

Proof. Fix a template vertex u. If $u \in N$, then these nodes are already assigned to world vertices, neither of which will be w_1 or w_2 and so $w_1, w_2 \notin C[u]$. Therefore $w_1 \sim_{c,u} w_2$. If $u \notin N$, and we have $((u, w_1), (v, x)) \in E_C$, then $(u, v) \in E_T$ and $(w_1, x) \in E_W$. v must be inside the node cover as those can be the only connections to u and therefore v must already be assigned to x. As $w_1 \sim_{N,M} w_2$, we must have $w_2 \in C[u]$ as well, and so must be joinable to the matching. This implies that $(w_2, x) \in E_W$. Hence we must have $((u, w_2), (v, x)) \in E_C$. Since this edge was chosen arbitrarily, we must have $w_1 \sim_{c,u} w_2$. Since this holds for all u, we must have $w_1 \sim_c w_2$.

On the other hand, if $w_1 \sim_c w_2$, for any template vertex t, if (t, w_1) is joinable to M, then (t, w_2) is also joinable to M. Hence, the template nodes for which w_1 and w_2 are candidates coincide, so that $w_1 \sim_{N,M} w_2$. □

Thus, until we have assigned a node cover, we can use candidate equivalence to prevent redundant branching; once we have matched all nodes in the node cover, we can check for node cover equivalence: a simpler condition.

The agreement of node cover equivalence and fully candidate equivalence is apparent in the candidate structure presented on the right of 2.1. From the candidate structure, the yellow nodes and the green nodes are fully candidate equivalent as they have the same neighbors, and that they are node cover equivalent as they only appear as candidates for the corresponding yellow and green nodes in the template.

The various notions of equivalence form a hierarchy. Structural equivalence of world nodes has the strictest requirements and implies all other forms of equivalence. Proposition 11 asserts that under mild conditions, full candidate equivalence and node cover equivalence are one and the same. We can also include candidate equivalence with respect to a template vertex as a weaker condition implied by full candidate equivalence that does not guarantee interchangeability. The following proposition summarizes these findings:

Proposition 12. *Suppose the assumptions of Proposition 11 hold. Given template vertex t, world vertices w_1, w_2, we have $w_1 \sim_s w_2 \Rightarrow w_1 \sim_{N,M} w_2 \Leftrightarrow w_1 \sim_c w_2 \Rightarrow w_1 \sim_{c,t} w_2$. Under the first three equivalences, w_1 and w_2 are interchangeable.*

2.3 Experiments for multichannel graphs

2.3.1 Tree search algorithm with equivalence

To demonstrate the utility of equivalence for the SMP, we adapt a state-of-the-art tree search subgraph isomorphism solver, Glasgow [54], using the modifications described in Algorithm **??**. We consider seven levels of equivalence: no equivalence (NE) (default), template structural equivalence (TE), world structural equivalence (WE), template and world structural equivalence (TEWE), candidate equivalence as in Proposition 6 (CE), full candidate equivalence as in Proposition 8 (FE), and node cover equivalence (NC). Each equivalence mode is integrated into the Glasgow solver separately. We assess the performance of our equivalence enhancements on the Glasgow solver, adapted to handle multiplex subgraph isomorphism problems. The adaptations involve minimal changes to the base algorithm, to ensure that matches are only made if they preserve the edges in every channel. To eliminate more candidates, we also perform a prefilter using the statistics and topology filters from [58] as well as maintain the subgraphs in each channel as the supplemental graphs used in the Glasgow algorithm.

2.3.2 Datasets and Experiment Results

We consider datasets including those from [59] and which represent both real world examples and synthetically generated data. The real world examples include a transportation network in Great Britain [26], an airline network [9], a social network built on interactions on Twitter related to the Higgs Boson [21], and COVID data [93]. For the transportation and twitter networks, the template is extracted from the world graph. The synthetically generated datasets are examples which represent emails, phone calls, financial transactions, among other interactions between individuals and are all generated as part of the DARPA-MAA program [42, 2, 17]. The subgraph isomorphisms to be detected may be a group of actors involved in adversarial activities including human trafficking and money laundering. The statistics regarding these different subgraph isomorphism problems are described in Table 2.1. For more details on these particular datasets, see [59].

The synthetic datasets are divided into three groups based on which organization generated the dataset: PNNL [17], GORDIAN [42], and IvySys Technologies [2].

For our experiments, we examine the seven modes of equivalence with a time limit of one hour to count as many solutions as possible. These experiments were run using our adapted version of the Glasgow solver. The amount of time required to enumerate all the solutions is displayed in Table 2.2 and the number of solutions found with a given method is displayed in Table 2.3 and 2.4.

Algorithm 2 Generic routine for a tree search with equivalance

```
1: function SOLVE(partial_match, cands)
2:     if MatchComplete(partial_match) then
3:         ReportMatch(partial_match)
4:         return
5:     ApplyFilters(partial_match, cands)
6:     Let $u$ = GetNextTemplateVertex()
7:     Let cands_copy = cands.copy()
8:     if Using World Equivalence then
9:         RecomputeEquivalence(partial_match, cands)
10:    Let ws = GenerateWorldVertices(cands, eq)
11:    for $v$ in ws do
12:        partial_match.match($u, v$)
13:        Solve(partial_match, cands_copy)
14:        partial_match.unmatch($u, v$)
15:        if Using Template Equivalence then
16:            for unmatched $u' \sim u$ do
17:                Set cands$[u', v] = 0$
18:    Let cands = cands_copy
19:    if Using World Equivalence then
20:        RestoreEquivalence()
21:    return
```

Table 2.1: Data on Multiplex Graphs from [87](© 2023 IEEE)

	Template		World		
Dataset	Nodes	Edges	Nodes	Edges	Chan.
Brit. Trans.	53	56	262377	475502	5
Higgs Twitter	115	2668	456626	5367315	4
Airlines	37	210	450	7177	37
PNNL RW	74	35	158	6407	3
PNNL v6-b0-s0	74	1620	22996	12318861	7
PNNL v6-b5-s0	64	1201	22994	12324975	7
PNNL v6-b1-s1	75	1335	22982	12324340	7
PNNL v6-b7-s1	81	1373	23011	12327168	7
GORDIAN v7-1	156	3045	190869	123267100	10
GORDIAN v7-2	92	715	190869	123264754	10
IvySys v7	92	195	2488	5470970	3
IvySys v11	103	387	1404	5719030	5
COVID	28	38	87580	1736985	9
Twitter - ER	5-15	4-31	456626	5367315	4

Table 2.2: Time (s) to enumerate solution spaces of multichannel problems. Experiments timed out at one hour. The Twitter-ER dataset is averaged over a collection of problems and timed out after ten minutes.This table is from [87](© 2023 IEEE)

Algorithm Dataset	CE	FE	NC	NE	TE	TEWE	WE
Brit. Trans.	3600	3600	3600	3600	3600	3600	3600
Higgs Twitter	3600	**369**	456	3600	3600	3600	3600
Airlines	0.34	**0.24**	1985	3600	1329	3600	3600
PNNL RW	3600	3600	3600	3600	3600	3600	3600
PNNL v6-b0-s0	41.6	42.1	41.8	**41.3**	41.7	41.4	41.6
PNNL v6-b1-s1	241	**240**	241	**240**	242	**240**	**240**
PNNL v6-b5-s0	62.6	58.1	58.9	**58.0**	62.3	62.3	63.0
PNNL v6-b7-s1	1133	200	201	3600	**188**	211	1138
GORDIAN v7-1	3600	**327**	3600	3600	3600	3600	3600
GORDIAN v7-2	3600	**316**	3600	3600	3600	3600	3600
IvySys v7	3600	3600	3600	3600	3600	3600	3600
IvySys v11	3600	3600	3600	3600	3600	3600	3600
COVID	3600	3600	3600	3600	3600	3600	3600
Twitter - ER	514.7	505.4	**494.8**	536.0	536.7	544.2	541.3

Table 2.3: Number of solutions found for multichannel problems within one hour (Part 1). The Twitter-ER dataset is averaged over a collection of problems and is timed out after ten minutes. This table is from [87](© 2023 IEEE)

Dataset	CE	FE	NC	NE
Brit. Trans.	1.48e+11	**2.34e+15**	4.97e+08	1.27e+07
Higgs Twitter	1.38e+14	**3.23e+14**	**3.23e+14**	5.65e+06
Airlines	**3.67e+09**	**3.67e+09**	**3.67e+09**	3.55e+09
PNNL RW	3.50e+09	2.78e+11	**4.72e+11**	8.59e+08
PNNL v6-b0-s0	**1.15e+03**	**1.15e+03**	**1.15e+03**	**1.15e+03**
PNNL v6-b1-s1	**1.15e+03**	**1.15e+03**	**1.15e+03**	**1.15e+03**
PNNL v6-b5-s0	**1.15e+03**	**1.15e+03**	**1.15e+03**	**1.15e+03**
PNNL v6-b7-s1	**3.14e+08**	**3.14e+08**	**3.14e+08**	8.57e+07
GORDIAN v7-1	1.11e+12	**9.13e+12**	1.35e+10	1.61e+07
GORDIAN v7-2	2.14e+11	**1.35e+16**	1.15e+15	1.72e+07
IvySys v7	2.04e+14	**8.04e+96**	2.09e+90	1.75e+09
IvySys v11	7.41e+10	**3.64e+89**	5.09e+66	1.77e+09
COVID	5.27e+14	**7.45e+21**	3.11e+20	9.63e+06
Twitter - ER	3.52e+08	8.84e+09	**1.40e+11**	7.17e+05

Table 2.4: Number of solutions found for multichannel problems within one hour (Part 2). The Twitter-ER dataset is averaged over a collection of problems and is timed out after ten minutes. This table is from [87](© 2023 IEEE)

Dataset	TE	TEWE	WE
Brit. Trans.	2.48e+12	2.02e+12	1.17e+07
Higgs Twitter	6.44e+06	5.72e+06	6.95e+06
Airlines	3.65e+09	8.11e+08	2.35e+09
PNNL RW	2.01e+10	5.00e+09	8.70e+08
PNNL v6-b0-s0	**1.15e+03**	**1.15e+03**	**1.15e+03**
PNNL v6-b1-s1	**1.15e+03**	**1.15e+03**	**1.15e+03**
PNNL v6-b5-s0	**1.15e+03**	**1.15e+03**	**1.15e+03**
PNNL v6-b7-s1	**3.14e+08**	**3.14e+08**	**3.14e+08**
GORDIAN v7-1	7.84e+09	5.34e+09	1.58e+07
GORDIAN v7-2	3.88e+08	3.19e+08	1.65e+07
IvySys v7	2.67e+47	5.84e+45	5.39e+07
IvySys v11	4.43e+72	6.64e+71	4.88e+07
COVID	9.53e+06	4.51e+07	1.31e+07
Twitter - ER	7.41e+05	6.14e+05	6.78 e+05

A quick inspection of the times illustrates that in a few cases (Airlines, GORDIAN, and Higgs Twitter), using full equivalence can enumerate the full solution space an order of magnitude faster than any other approach. This speedup is reflected in the solution count table for which FE finds significantly many more solutions. The other methods only find a mere fraction of the total solutions. The NC method often appears to be the second best both in terms of solutions found and time taken to enumerate all. This makes sense given Proposition 12. TE appears to be the third best method which can be explained by the simplicity of implementation and having no need to recompute equivalence. WE and CE are not competitive with the other methods. The datasets bear different qualities that illustrate why certain levels of equivalence work better than others. We discuss a few datasets in detail.

2.4 Venn-Diagram representation of condensed solution

When there are large equivalent classes in the template graph, the intersection of their candidate sets are very common. On the other hand, their candidate sets often satisfy the definition of node cover equivalence. This is because a large equivalent class in the template is usually comprised of low degree nodes, especially those with degree one. If we include the common neighbors of low degree nodes in the node cover, then the candidates of equivalent classes will satisfy the definition of node cover equivalence.

The equivalent classes both in the template graph and world graph lead to a combinatorial complexity when solving the all-different assignment problem for counting the total number of solutions. To deal with this case, we can visualize the intersection of candidate sets with a venn diagram. We consider the node cover equivalence with intersection after we get a partial match M on the node cover.

Definition 13. *Let M be a partial match M on a node cover N of V_T and suppose that for all $u \in V_T \setminus N$, for two nodes w_1, w_2 in the world graph, they are equivalent to each other* *as long as they are the candidate of the same template nodes. That is to say,* $w_1 \sim_v w_2 \Leftrightarrow \{u|w_1 \in C[u]\} = \{v|w_2 \in C[v]\}$

The difference of the node cover equivalence with intersection with candidate equivalence is that this equivalence does not change when we perform the tree search as long as the partial match is given.

Actually, if we draw the Venn diagram of the candidate sets of each node outside the partial match, then the nodes in the same section of the Venn diagram are equivalent to each other. This is the visualization of the nodes that has node cover equivalence.

If we have L unfixed nodes, we can store the size of each equivalence class in a list of length 2^L. We call this list **Venn diagram List**. In practice, this list is often very sparse. Then we use the following tree search to extend the partial match 3 to a complete representative solution. The way to do this is to choose the corresponding numbers from each section of the Venn diagram such that their sums match the size of template equivalent classes. The number of nodes to choose is stored in a **solution list**.

In the following sections, we will show some numerical examples and the venn-diagram representation of condensed solution from which we can generate large numbers of trivial solutions through node cover equivalence combined with other equivalence relations.

2.4.1 IvySys

The Ivysys template and world graphs [2] are separately generated to match the degree distribution and email behavior of the Enron email dataset and have the most complex solution space. None of the methods were successful at enumerating all solutions. The vastness of the solution space is in contrast to the size of the graphs which only have thousands of nodes. The complexity emerges from the preponderance of template leaf nodes as shown in figure 1.3, depicting one solution class from which 7.82×10^{103} solutions may be generated. Figure 2.2 depicts the compressed representation of the world subgraph for this solution as well as

Algorithm 3 Extending the Partial Match using Venn diagram

```
1: function EXTENSION(partial_match,template_eq_class_size venn_diagram_list,solution_list)
2:     if ExtensionComplete(partial_match,solution_list) then
3:         ReportMatch(partial_match,solution_list)
4:         return
5:     Let $u$ = GetNextTemplateVertex()
6:     Let venn_diagram_list_copy = venn_diagram_list.copy()
7:     Let solution_list_copy = solution_list.copy()
8:     Let ws = GenerateSolutionSequence( template_eq_class_size, venn_diagram_list_copy)
9:     for $v$ in ws do
10:        venn_diagram_list-=v
11:        solution_list+=v
12:        Extension(partial_match,template_eq_class_size, venn_diagram_list_copy, solution_list_copy)
13:    return
```

a Venn diagram displaying candidates of certain template nodes. The TE solver finds an astonishing 10^{47} solutions for IvySys v7. However, using the FE method still dramatically increases the solution count, by mapping these large template equivalent classes into larger world equivalence classes. An equivalence-informed subgraph search is essential as the NE method finds only 1.75×10^9 solutions, 90 orders of magnitude less than the FE search. Furthermore, a typical subgraph search would assign each group of leaf nodes sequentially meaning only the candidates of the last group would be explored. Incorporating symmetry gives a fuller vision of the solution space.

2.4.2 COVID

We lastly apply our algorithm to the problem of querying a knowledge graph representing known causal relations between a large variety of biochemical entities. This problem arises from a desire to extracting causal knowledge in an automated fashion from the research literature. In [93], a knowledge graph is assembled from multiple sources including the COVID-19 Open Research Dataset [83], the Blender Knowledge Graph [84], and the comparative toxigenomics database [20]. The authors of [93] then create a query representing how SARS-CoV-2 might cause a pathway leading to a cytokine-storm in COVID-19 patients, but is generalized to detect other possible confounding factors in the pathway.

When rephrased as a multichannel subgraph isomorphism problem, template and world nodes represent biochemical entities. Some template nodes are specified, and others are labeled as a chemical, gene or protein. The 9 channels in this problem are various known types of interactions between entities, e.g., activation. A solution is an assignment of each node which has the desired chemical interactions.

Figure 2.3 depicts the template and Venn diagrams of candidates sets for one solution class and exposes unspecified template nodes with a large amount of candidates. Such information is useful to an analyst for determining confounding factors in a pathway and suggesting label information or interactions to add to better specify the entire solution space.

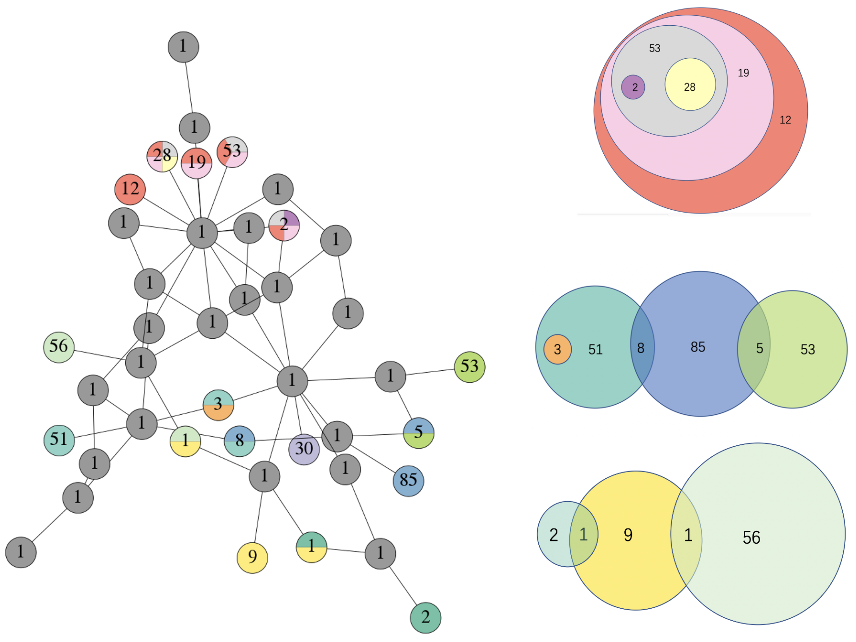

Figure 2.2: IvySys v7 [2] Compressed solution-induced world graph (left) and the Venn diagram representation of intersecting candidate sets in world graph(right) for a solution class from which 7.82×10^{103} solutions can be generated. The number in each section in the Venn diagram represents the size of a node cover equivalence class in the world graph. All solutions represented by this compressed solution can be generated by mapping each colored node in the template to the set in the Venn diagram with the same color. This graph is from [87](© 2023 IEEE)

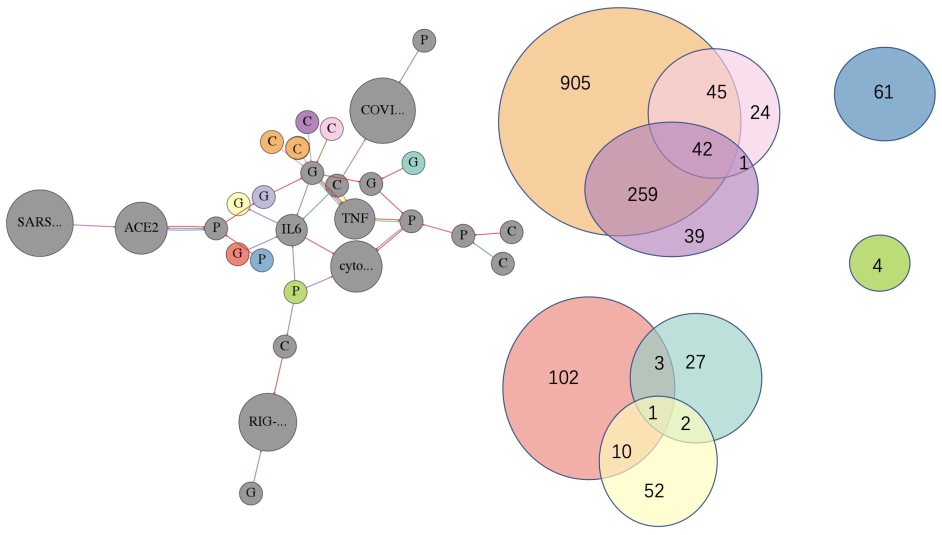

Figure 2.3: COVID-19 [93] template (left) and the Venn diagram of candidate sets in world graph (right) from which 2.6×10^{18} solutions can be generated in one solution class. Each section in the Venn Diagram represents a node cover equivalence class, and the number in the section is the size of the class. A few template nodes were specified at the start whereas others simply received a node label of C, P, or G indicating chemical, protein, and gene respectively. The solutions may be generated by mapping non-gray template nodes of one color to world nodes in the Venn diagram section of the same color. This graph is from [87](© 2023 IEEE)

CHAPTER 3

Active learning for Subgraph matching

This chapter is about reducting the solution space with the help of active learning. The work was published in [30] where I am the first author. In this chapter, we present different strategies to fixing the candidate nodes for a small number template nodes and we ask the question - which template nodes should be chosen so as to reduce the complexity of the solution space the most? I contributed to these strategies including one original entropy based strategy called edge entropy. This strategy performs well for some difficult datasets like PNNL real world and British transportation.

3.1 background for active learning

In Chapter 1, we introduced the application of the subgraph matching to real life problems where we would need to find all the subgraph matching solutions. On the other hand. in some real use-case scenario, one might need to identify one specific subgraph isomorphism by restricting the candidate nodes/edges in the world graph. There are a number of reasons why this would be - for example if the knowledge graph represents data related to an investigation involving an unknown actor, such as in a homicide investigation or a serial offender, it would be important to identify the actual person involved. The consequences of misidentifying someone could be grave - both for the person wrongly identified and for potential future victims of the actual person involved. In some cases there could be more than one subgraph isomorphism of relevance, for example in the case of identifying different but equally important pathways in a biochemical reaction network [29] or the case of

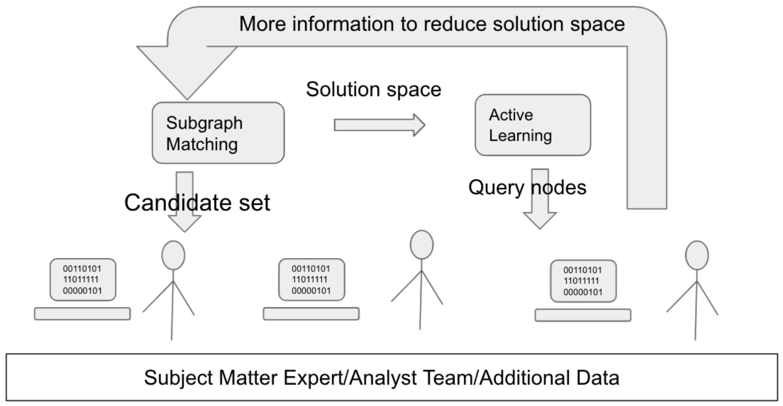

Figure 3.1: Active learning flowchart for subgraph matching [30](© 2021 IEEE). A subgraph matching algorithm determines all potential candidates for template nodes (using constraint propagation). An active learning algorithm determines the optimal nodes for SMEs to obtain additional constraints/information. This is fed back into the subgraph matching algorithm.

identifying groups involved in human trafficking or smuggling. Likewise, organizations or people interested in identifying those wrongly accused of crimes could look at a knowledge graph of information that might present alternate scenarios. In a real life setting, this could entail additional constraints added to the problem space such as attributes for the nodes (e.g. names, dates, times etc). It could also involve addition of more data. Such information might come at a cost and therefore it would be of interest to understand strategies to reduce the complexity of the solution space with the minimal cost. A flowchart describing how this approach might be used in a real world setting in shown in Fig. 3.1. We obtain the candidate set for all template nodes using a filter based subgraph matching algorithm. Then, an active learning algorithm determines the optimal nodes for subject matter experts to obtain additional information. After that, the additional information is fed back into the subgraph matching algorithm to get reduced candidate sets. The need for expert input for multichannel subgraph matching is also illustrated quite well by the benchmark datasets developed under the DARPA MAA (Modeling Adversarial Activity) program [69]. The theme of this program involves template graphs that describe a series of actions and the world graph constructed from relevant data. The necessity of involving input from a human expert to this problem inspires active learning.

Active learning is an area of research in statistical machine learning that involves a subject matter expert (SME) in the actual algorithm for classification of points in a dataset. Supervised machine learning algorithms require an abundance of labeled data. In the real-world however, unlabeled data is common and accurate labeling may require human involvement that can not be crowd-sourced due to privacy or security reasons. Semi-supervised methods use significantly fewer training points. Meanwhile, the choice of labelled data often affects classifier performance. Active learning involves the use of an algorithm or formula to choose individual data points for labeling by a SME. Then the newly labeled data are included in the semi-supervised learning problem. These active learning methods iterate between the following procedures: (1) Training a model given the current labeled data (2) Choosing one or a batch of query points in the unlabeled set based on an active learning criterion such as an acquisition function. Most active learning acquisition functions for statistical machine learning belong to one of a few categories: uncertainty [71, 36, 25], margin [77, 4, 38], clustering [19, 49], and look-ahead [92, 8].

Researchers have introduced active learning into problems similar to subgraph isomorphism problems such as the network alignment problem. This problem tries to find an optimal mapping of graph nodes with maximum similarity between the nodes and edges, in which a cost function measures differences between the nodes or edges. The optimal solution with least cost is given by updating the probability distribution for each node. Prior research shows that better alignment can be achieved by introducing interaction with a human to obtain extra information on certain nodes. For example, in [70], researchers compare three probability matrix based query strategies. In [15], the cost function of network alignment is updated after interaction with human. In [51], the authors examine the case where experts only provide partial information about the mapping of certain nodes instead of the exact answer. In [64], the authors study vertex nomination in the inexact subgraph matching problem. There is not much research on the active learning algorithms for subgraph isomorphism problem. However, their need can be both justified by the necessity of involvement of human experts as well as the active learning research on the similar problems.

One might have the objective to identify one specific subgraph isormorphism by restricting the candidate nodes/edges in the world graph. There are a number of reasons why this would be - for example if the knowledge graph represents data related to an investigation involving an unknown actor, such as in a homicide investigation or a serial offender, it would be important to identify the actual person involved. The consequences of misidentifying someone could be grave - both for the person wrongly identified and for potential future victims of the actual person involved. In some cases there could be more than one subgraph isomorphism of relevance, for example in the case of identifying different but equally important pathways in a biochemical reaction network or the case of identifying groups involved in human trafficking or smuggling. Likewise, organizations or people interested in identifying those wrongly accused of crimes could look at a knowledge graph of information that might present alternate scenarios. In a real life setting, this could entail additional constraints added to the problem space such as attributes for the nodes (e.g. names, dates, times etc). It could also involve addition of more data. Such information might come at a cost and therefore it would be of interest to understand strategies to reduce the complexity of the solution space with the minimal cost.

3.2 Active learning assisting reduction of solution space

Below we propose a few simple querying strategies for active learning to reduce the solution space. The querying strategies are carried out after first running the constraint propagation algorithm to determine a potential list of candidate nodes. In numerical examples in this chapter we choose simple filtering strategies without extensive tree searches. Thus we are not solving the SNSP or MCSP in full, rather providing a pared down list of candidates under consideration for the subgraph matching problem. The reason for this is that an active learning method requires code that can run in real time for analysts and this will be essentially guaranteed for the constraint filters but not for extensive tree searches to validate all the candidates.

3.2.1 Querying strategies for template nodes

This chapter focuses on querying strategies for template nodes. These are the easiest to analyze and visualize by displaying the candidate counts for each template node, with the templates being small enough that they can be displayed simply in a two dimensional diagram. Such a strategy is also important for SME/analysts to interact with the active learning algorithms. Below we present several strategies for choosing template nodes to query.

3.2.2 Local template-based strategies

First we consider two simple strategies:

- choose the template nodes with the largest degree centrality measure (number of edges connecting that node)
- choose the nodes with largest sum of the number of candidates for neighboring template nodes

3.2.3 Edge entropy

We introduce a notion of “edge entropy”. One purpose of the query is to simplify the complex part of the graph to enable less costly tree searches. Shannon’s entropy is one tool to measure complexity. In the subgraph matching problem, the mapping of an edge is usually more complex than the mapping of a node. We define the following edge entropy:

$$-\sum_i p_i log(p_i), \tag{3.1}$$

where p_i is the probability that the mapping of an edge to its candidate set passes the local filter in the affected region. Here i is summed over all edges connected to the node in question and the entropy measure is assigned to that node.

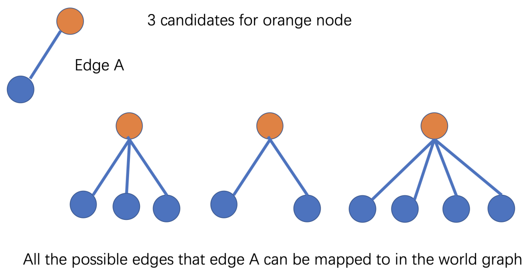
3 candidates for orange node
Edge A
All the possible edges that edge A can be mapped to in the world graph

Figure 3.2: Edge entropy toy example. This figure is from [30](© 2021 IEEE)

Fig. 3.2 shows a toy example, in which we want to query information for the orange node, connected to edge A. In the world graph, this edge can be mapped to nine possible candidate edges. There are three cases that the orange node can map to. In the first case, there are three edges connected to the selected world node. So the probability in this case is 1/3. Similarly the probability for the remaining edges are 2/9 and 4/9 So the edge entropy of this edge is $-(\frac{1}{3}log(\frac{1}{3})+\frac{2}{9}log(\frac{2}{9})+\frac{4}{9}log(\frac{4}{9}))$. For each template node, we calculate the sum of edge entropies of all the edges that are incident to the template node.

While pruning the candidate list, we may run into cases where the size of the candidate set is too large. In this case, we are unlikely to find all the subgraph isomorphisms using tree search because of limited computation time and resources. By introducing active learning in the pruning of candidate list, we can query information about a certain nodes or edges in the template or world graph to reduce the candidate list to an acceptable size. The problem of determining the nodes and edges to query is the active learning problem in subgraph isomorphism problem. Below we show some examples using datasets for multichannel networks.

3.2.4 Ivysys V7 - sum of candidates

We show an example from IvySys Version 7 [2], developed by Ivysys technologies for the DARPA MAA program, with three channels corresponding to financial, communication and logistics transactions. This dataset has a template with 92 nodes and 195 edges. The world graph has 2,488 nodes and 5,470,970 edges. To date the entire solution space has not been solved for, although a representative solution with over 10^{100} isomorphisms is identified in [87]. The template has a tree-like sparse structure, resulting in no unique candidates after applying different levels of filtering methods, as seen in Fig. 3.3 (left).

We can significantly reduce the solution space by querying key nodes, selected according to the maximum sum of neighboring candidates (see Fig. 3.3 (right - the nodes are dark green)). We note that the degree centrality metric identifies five out of six of the same query nodes in this example. The edge entropy criterion identifies the same six nodes but with a different ordering. The sixth one is the query node that does not have many leaves. Based on the query from the active learning criteria, we specific world nodes to the queried template nodes. We chose world nodes from the first isomorphism found by the code from [87], as a proxy for additional information supplied by SMEs. After fixing the queried nodes, the remaining candidates in the center of the template are reduced significantly, mostly to a single world node or a few world nodes. This example has a network structure reminiscent of core-periphery structure [67]. However, the leaf nodes connecting to the core nodes still have many candidates. Due to large equivalence classes in the template and world graph [87, 61], the number of solutions for the SMP problem is still huge. That said, one can still obtain useful information for SMEs and analysts with a Venn diagram representation of candidates for several equivalence classes as shown in Fig. 3.3 in the bottom row. It shows the intersection of candidate sets of the largest three equivalent classes in the template. The difficulty of the subgraph isomorphism comes from the alldifferent problem [65] in assigning

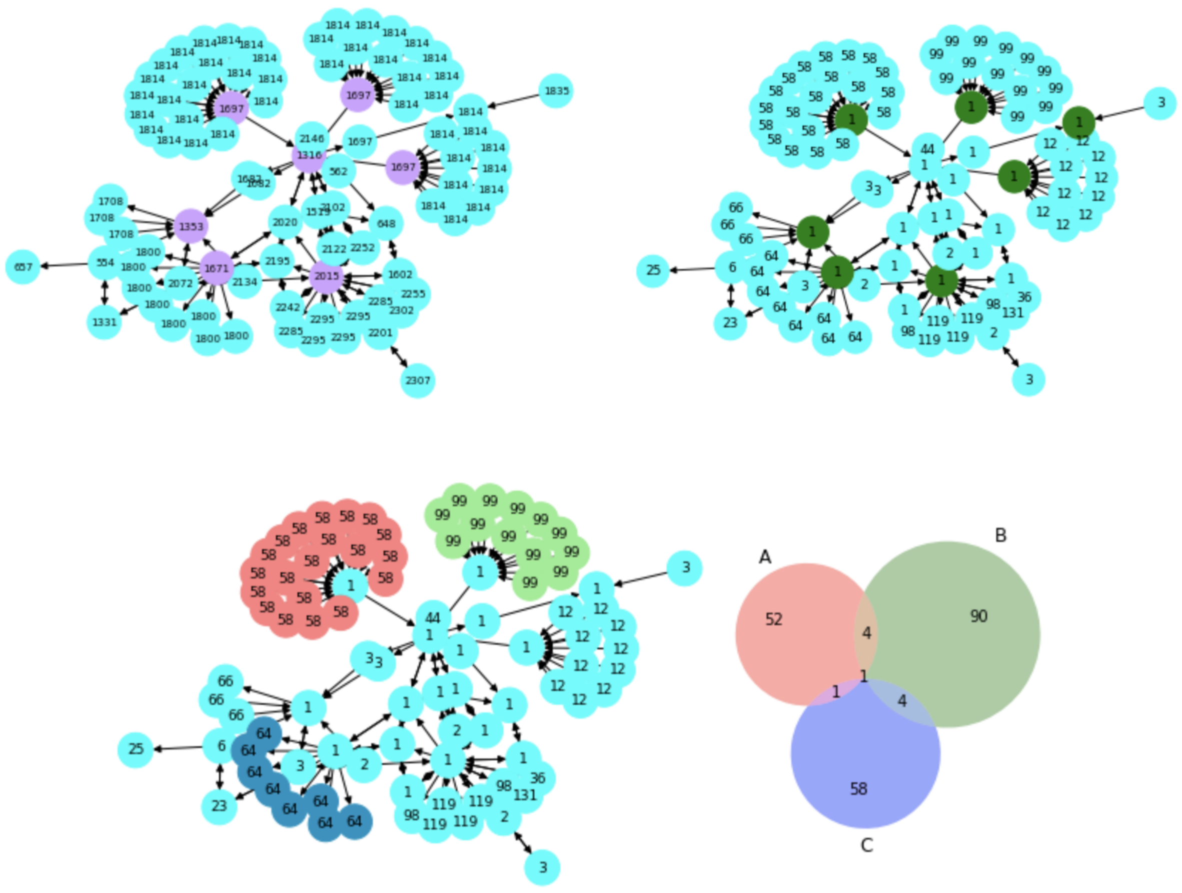

Figure 3.3: (Top left) Number of candidates for querying the nodes of ivyvys v7 template after implementing the main filters in [87]. The template nodes with the highest degree centrality are marked in purple. This figure is from [30](© 2021 IEEE) (Top right) Number of candidates after querying the template nodes with maximum sum of neighboring candidates, using the first isomorphism in the first found representative solution using the code in [87]. (bottom) The overlapping structure of candidate sets after the query. The circles in the Venn diagram represent the candidate sets of the nodes in the template with corresponding color - Set A (red nodes), Set B (green nodes), Set C blue nodes).

the template nodes to its candidate set. But analysts can get an idea of what the solution space looks like before solving the all-different problem. The Venn Diagram itself could be incorporated into a computational tool in which an analysts could click on a portion of the Venn diagram to obtain an itemized list of those candidates.

3.2.5 Example from PNNL real world

This dataset was made by Pacific Northwest Nationabl Lab from a social media dataset collected by Matteo Magnani and Luca Rossi [13, 50]. It involves friend/follower relationships on three social media platforms, each of which corresponds to a channel. The template graph has 35 nodes and 158 edges and is an *induced subgraph* in the world graph.The world graph has 6,407 nodes and 74,862 edges in six channels. Additional SI solutions can be found with additional edges. We use the induced subgraph as "ground truth" for our query analysis. We refer to this dataset and its induced subgraph template as the "PNNL Real World dataset". An analysis in [59] found a total of 2.12×10^{12} isomorphisms using all filters including the elimination filter which performs a final tree search. Fig. 3.4 shows the template for this subgraph matching problem in which each node has listed the number of candidates from the world graph after applying the node level statistics, topology, repeated-set, and neighborhood filters (but not elimination filter) [59]. For this reason the candidate counts are slightly higher than what are shown in [59] and are more realistic for an active learning scenario when there may not be sufficient time to run extensive tree searches, especially when additional information can be added to greatly reduce the solution space. The entropy values of each of the template nodes are shown in the top right. The full candidate count is shown in the bottom figures for two choices of queries - on the left the top two entropy node are chosen. On the right the third and fourth highest entropy nodes are chosen. The choice of the highest entropy nodes clearly has a much smaller solution space than the resulting solution space for the alternate choices. We also tried the sum of neighboring candidate counts as a metric and found that we needed to query the top three nodes with that metric

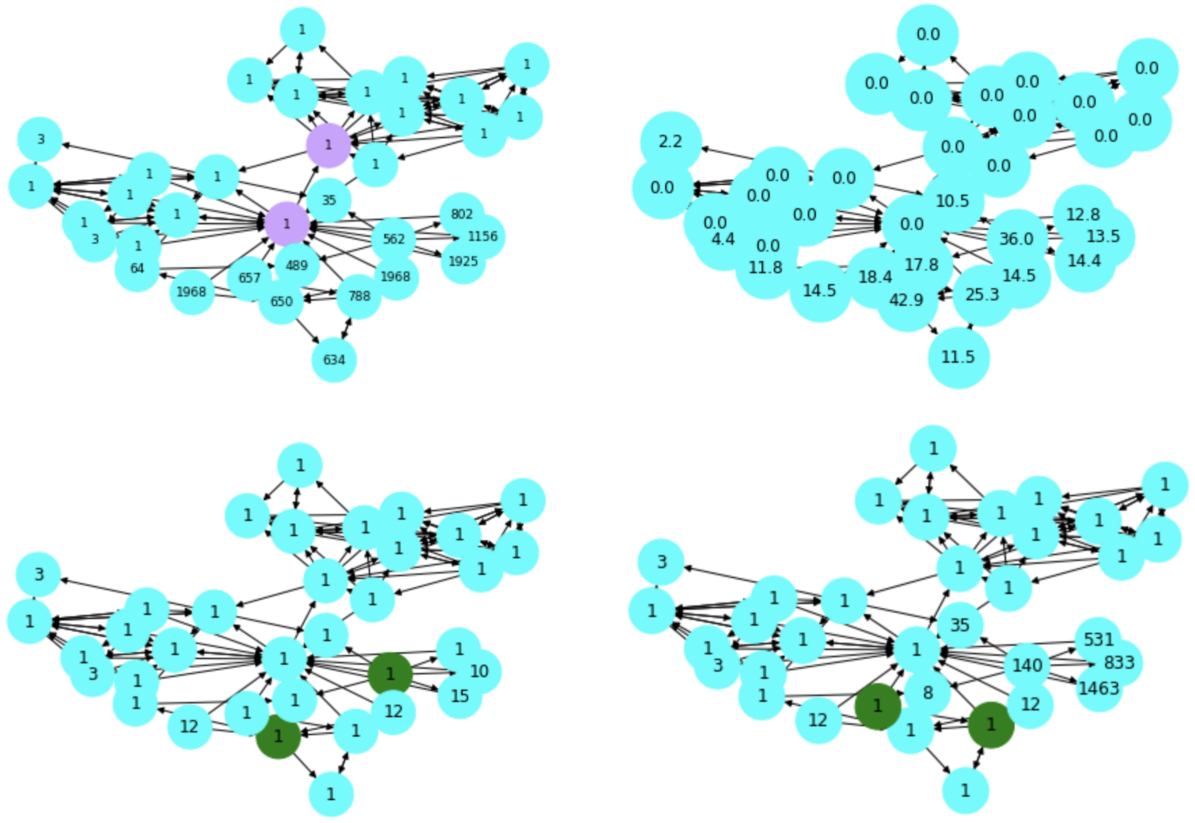

Figure 3.4: (Top left) Number of candidates of PNNL real world template, after running basic filters [30](© 2021 IEEE). The template nodes with the highest degree centrality are marked in purple. Notice that they each only have one candidate node and are thus not useful to query in an active learning scenario. (Top right) Entropy values for each each template node. (bottom left) Number of candidates after querying the two nodes with highest entropy. (Bottom right) Number of candidates after querying the the nodes with third and fourth highest entropy rather than the top two highest entropy.

to get the same reduction of the solution space as what was found by querying the top two nodes according to the entropy metric.

CHAPTER 4

Active learning to locate one solution

This chapter is based on the manuscript “Iterative active learning strategies for subgraph matching” by myself, Dominic Yang, and my advisor Andrea Bertozzi. I am the lead author on this paper and this chapter summarizes my contributions. Dominic Yang contributed to studies on single channel networks.

4.1 Definitions and Terminology

In the first chapter, we introduce the definition of a multichannel graph and the subgraph matching problem (SMP) for multichannel graph. We also introduced the notion of edge preserve mapping (EPM) and subgraph isomorphism (SI). This section provides the intuition and algorithm for our active learning framework. as well as formalizes the theoretical problems associated to this framework.

4.1.1 Active Learning Framework

The problem we are interested in is to solve the SMP while simultaneously ruling out SIs that can be eliminated by additional information that is available or potentially available to subject matter experts (SME). The end goal is to have a final solution to the SMP, after elimination of extraneous SIs, that has a modest solution count and provides a final list of SIs that are clearly of interest to the the application problem. SMEs are part of the active learning procedure, providing information based on active learning queries.

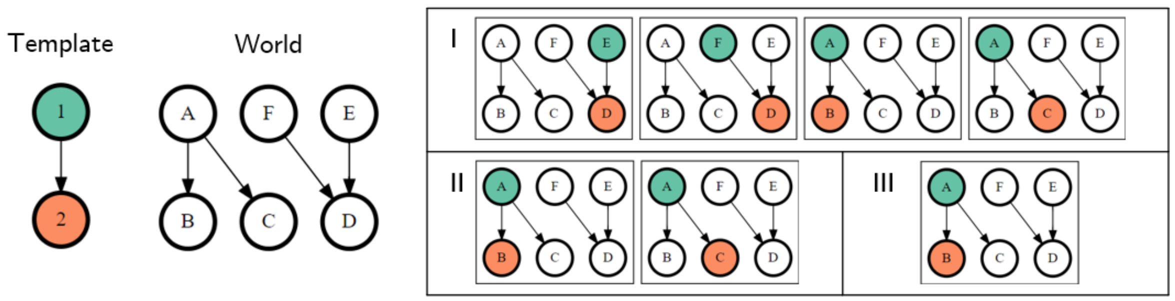

Figure 4.1: Solution spaces for example template and world graph following active learning queries. The ground truth subgraph matching is given by mapping 1 to A and 2 to B. In I, there are initially four possible SIs, in II, there are two SIs after querying template vertex 1 and finding it maps to A, and in III, finally we have reduced the solution space to one SI after querying vertex 2 and finding it maps to B.

In this subsection, we will present the general algorithmic framework we will be using to test various active learning strategies. For each experiment, we will assume we have a template graph G_t, world graph G_w, and a valid subgraph matching f which we are trying to determine. $f(t)$ will represent the ground truth world vertex which is associated with a given template vertex t.

As a toy example of how the active learning problem may proceed, we consider the example template and world presented in Figure 4.1. Initially, there are four possible subgraph matchings in the world graph and the ground truth matching maps nodes 1 and 2 to A and B, respectively. At this stage, we query template vertex 1 and determine its true assignment, A. From this knowledge, we can rule out any solution which maps 2 to a world vertex not adjacent to A eliminating two solutions. Then, with two solutions left we query template vertex 2 finding it maps to B which fully determines the true subgraph matching after two queries. However, if we instead had queried template vertex 2 first, we would have found it mapped to B, which only one of the candidate solutions does, and we can determine the solution in 1 query. From this simple example, we observe that having a clever strategy for querying template vertices is important to minimizing the total work needed to find a solution.

During our subgraph search, we will encode the full set of information in **candidate sets**, which are sets of world vertices to which a given template vertex $t \in V_t$ may be mapped. We denote the candidate set for template vertex t by $C(t)$. They exclude any world vertex which has been ruled out as a candidate for t.

The process of ruling out candidates is known as **filtering** and makes use of basic properties of subgraph isomorphisms known to be true (e.g., that template vertices must map to nodes of higher degree or that neighboring template vertices map to neighboring world vertices). The choice of the exact filters to use to prune candidates is an active area of research with many possible choices of filters (see [79, 73, 54, 6, 57] for a selection of papers discussing a variety of filtering choices).

We now present in Algorithm 4 the basic framework by which we perform active learning for the subgraph matching problem. The algorithm proceeds by alternately filtering the current candidate sets of each template vertex based on the filtering criteria described in [59] and querying for a template vertex according to a given strategy.

Algorithm 4 Active Learning Template Query Loop

```
Input: Template $G_t$, World $G_w$, Matching $f$
$C \leftarrow$ Filter($G_t$, $G_w$)                              $\triangleright$ Initialize domains
$count \leftarrow 0$
while Not all $t$ in $V_t$ have 1 candidate do
    $t \leftarrow$ QueryStrategy($G_t$, $G_w$, $C$)
    Match($t$, $f(t)$)
    $C \leftarrow$ Filter($G_t$, $G_w$, $C$)
    $count \leftarrow count + 1$
Return $count$
```

The QueryStrategy function picks out a template vertex to query and is the central object of study in this chapter and we will discuss it in detail in Section 4.3. The Match function formally associates t and $f(t)$, and then the Filter function eliminates candidates based on the various filters. Once filtering has reduced the size of the candidate sets of each template vertex to one candidate, the matching has been determined, and we report the number of queries made.

We now formally state the problem. Our thesis will address:

Definition 14 (Optimal Template Query Problem). *Given G_t, G_w, candidate sets $C(t)$ for $t \in V_t$, and a fixed choice of filter, which query strategy will require the fewest template queries as given by Algorithm 4?*

In this thesis, our metric is the number of queries required to fully determine a subgraph matching. We envision this as the most expensive operation as it potentially requires expert involvement to perform the query. In our problem, each template node requires the same amount of work to query, but future work may study an extension of this problem which varies the work required on a per-node basis. We also note that this problem depends on the choice of filter. Stronger filters (filters which eliminate more candidates) will give us more information and it is possible that strategies which make better use of this information may perform better in that context. We will sidestep this problem instead opting to fix the choice of filter to that of the LAD filter introduced in [73].

4.1.2 Associated Theoretical Problems

Within this framework, we can consider two similar theoretical problems given a template G_t and a world G_w. The first problem involves a candidate solution $f \in F(G_t, G_w)$, and we wish to determine the minimal number of template vertices we need to query to verify that f is the true solution. This is formalized in the following definitions:

Definition 15 (Solution Verifying Set). *Given G_t, G_w and $f \in F(G_t, G_w)$, a* ***solution***

verifying set *is a subset $A \subset V_t$ such that if we have $g \in F(G_t, G_w)$ with $g(t) = f(t)$ for all $t \in A$, then $g = f$.*

Definition 16 (Minimal Solution Verifying Set Problem)**.** *Given G_t, G_w and $f \in F(G_t, G_w)$, what is a solution verifying set for f of minimal size?*

In the example in Figure 4.1, we observe that for any of the solutions, we need only to query one template vertex to verify a given solution. For solutions 1 and 2, we query vertex 1 and for solutions 3 and 4, we query vertex 2.

For the second problem, we do not have a candidate solution. Rather, we wish to know the minimal size of a subset of V_t for which if we knew the images of these vertices, we could uniquely identify the images of the remaining vertices. We introduce the following definition and problem to this end:

Definition 17 (Determining Set)**.** *Given G_t and G_w, a* ***determining set*** *is a subset $A \subset V_t$ where for every $f, g \in F(G_t, G_w)$ if $f(t) = g(t)$ for $t \in A$, then $f = g$.*

Definition 18 (Minimal Determining Set Problem)**.** *Given G_t and G_w, what is a determining set of minimal size?*

For the problem in Figure 4.1, we need to query both template vertices 1 and 2 to determine the subgraph isomorphism in all cases. If we query template vertex 1 and 1 maps to world vertex C, there are still two SIs which are possible. Similarly, if we query template vertex 2 and find that 2 maps to world vertex D, there are also two possible SIs. Hence, the minimal determining set is $\{1, 2\}$ and is of size 2. Note that this is a counterexample to the statement that the size of the minimal determining set is the maximal size of a minimal verifying set over all SIs $f \in F(G_t, G_w)$.

The decision variants of both of these problems are NP-complete. We will prove that the solution verification set problem is NP-complete in Section 4.2 by reduction from the minimum set cover problem. As for the minimal determining set problem, we note that in the case that $G_t = G_w$, finding a determining set of a certain size is equivalent to finding a base for the automorphism group of G_t, which is already known to be NP-complete [7].

In practice, the optimal template query problem is intermediate to both of these theoretical problems as we will generally not have a solution f at our disposal (and almost certainly not the whole set of SIs $F(G_t, G_w)$). However, we may have some prior information and in the process of querying template vertices, we will be gathering additional information which will aid us in determining the ground truth SI.

4.2 NP-Completeness of the Minimal Solution Verification Set Problem

In this section, we will present results on the complexity of the optimal template query problem. We first introduce the set cover problem, which is well known for being NP-complete. Then we prove that the minimum set cover problem is reducible to the minimal solution verifying set problem proving NP-completeness of this problem. This shows that the optimal template query problem is at least as hard as solving the minimum set cover problem being an extension of the minimal solution verifying set problem.

4.2.1 Reduction of Minimum Set Cover to Minimum Solution Verification Set

We define the minimum set cover problem as follows:

Definition 19 (Minimum Set Cover Problem)**.** *Suppose S is a set $S = \{1, 2, \ldots, m\}$, and we have k subsets $S_i \subseteq S$, $1 \leq i \leq k$. The **minimum set cover problem** is to find an index set $I \subseteq \{1, 2 \ldots, k\}$ with minimum cardinality, such that $S \subseteq \cup_{i \in I} S_i$.*

We assume the problem is non-trivial and has at least one set cover, i.e., $\cup_{i=1}^{n} S_i = S$. We denote the members of each subset $S_i = \{a_1^{(i)}, a_2^{(i)}, \ldots, a_{r_i}^{(i)}\}$.

The associated decision problem (i.e., determining if there is a set cover with fewer than

N elements) is NP-complete [41] and therefore believed to be unsolvable in polynomial time with k. We will prove the solution verification set problem is NP-complete via reduction from the minimum set cover problem.

Given an instance of the minimum set cover problem $(S, \{S_1, \ldots, S_n\})$, we will produce an equivalent instance of the solution verification set problem. Let $V_t = \{v_1, v_2, \ldots, v_k\}$ be the vertices of complete single channel simple graph with k nodes. That is to say

$$E_T(v_i, v_j) = \begin{cases} 1 & i \neq j \\ 0 & \text{otherwise} \end{cases}.$$

We impose that each node has a unique label: $L(v_i) = i$. For each set $S_i = \{a_1^{(i)}, a_2^{(i)}, \ldots, a_{r_i}^{(i)}\}$, We define its associate vertex set $V_i = \{v_{a_1}^{(i)}, v_{a_2}^{(i)}, \ldots v_{a_{r_i}}^{(i)}\}$ and ground truth vertex v_{gt}^i. The labels of these vertex are equal to i. That is to say $L(v) = i \Leftrightarrow v \in V_i \textit{ or } v = v_{gt}^i$. The vertices of world graph are defined to be the union of all the vertex sets and ground truth vertices:

$$V_w = \cup_{i=1}^k (V_i \cup \{v_{gt}^i\}).$$

We also define a mapping f on the world vertices:

$$f : V_w \longrightarrow \mathcal{P}(S \cup \{gt\})$$
$$f(v_{a_j}^{(i)}) = \{a_j^{(i)}\}$$
$$f(v_{gt}^i) = S \setminus S_i \cup \{gt\}.$$

where $\mathcal{P}(A)$ is the power set of A. The edges in the world graph are defined as followed:

$$E_W(u, v) = \begin{cases} 1 & f(u) \cap f(v) \neq \emptyset \\ 0 & f(u) \cap f(v) = \emptyset \end{cases}.$$

The ground truth is set to be: $SI(v_i) = v_{gt}^i$.

Theorem 20. *Using the above definitions, g is a subgraph isomorphism from the template graph G_t to the world graph G_w if and only if:*

$$g(v_i) \in V_i \cup \{v_{gt}^i\} \textit{ and } \cap_{i=1}^k f(g(v_i)) \neq \emptyset \tag{4.1}$$

Proof. ($\Longrightarrow$) Let g satisfy (4.1). From the definition, we know if $i \neq j$, then $V_i \cap V_j = \emptyset$. Then $g(v_i) \in V_i \cup \{v^i_{gt}\}$ implies if $i \neq j$, $g(v_i) \neq g(v_j)$. So g is an injection. Second, $\cap^k_{i=1} f(g(v_i)) \neq \emptyset$ implies if $i \neq j$, then $f(g(v_i)) \cap f(g(v_j)) \neq \emptyset$. By definition, $E_W(g(v_i), g(v_j)) = 1$. So g is edge-preserving. Finally, $L(g(v_i)) = i$, so g preserves labels. So g is a subgraph isomorphism

($\Longleftarrow$) Let g be a subgraph isomorphism. Then g preserves labels. So $L(g(v_i)) = i$, $g(v_i) \in V_i \cup \{v^i_{gt}\}$. If $gt \in \cap^k_{i=1} f(g(v_i))$, then the conclusion is true. If not, then $\exists p, gt \notin f(g(v_p))$ That means $g(v_p) = v^{(i)}_{a_q}$ for some $a^{(i)}_q \in S$. So $f(g(v_p)) = \{a^{(i)}_q\}$ has only one element. Since g is edge-preserving, if $s \neq p$, then $E_W(g(v_s), g(v_p)) = 1, f(g(v_p)) \cap f(g(v_s)) \neq \emptyset$. So $a^{(i)}_q \in f(g(v_s))$ Thus $\cap^k_{i=1} f(g(v_i)) \neq \emptyset$.

□

Theorem 21. *Suppose S has a set cover $S \in \cup_{i \in K} S_i$, where $K \subset \{1, 2, \ldots, k\}$, then $\cup_{i \in K}\{v_i\}$ is a valid active learning query. The converse is also true.*

Proof. For any subgraph isomorphism g, suppose g has the same image as the ground truth SI for the queried nodes, i.e. $\forall i \in K$, $g(v_i) = v^i_{gt}$. Since $f(SI(v_i)) \cap S_i = \emptyset$, $S_i \notin \cap^k_{j=1} f(g(v_j))$ Thus $\cup_{i \in K} S_i \notin \cap^k_{j=1} f(g(v_j))$. Since $S \in \cup_{i \in K} S_i$,We have $\cap^k_{j=1} f(g(v_j)) = \{gt\}$. Thus $\forall j, g(v_j) = SI(v_j)$. Thus $g = SI$.

On the other hand, suppose$\cup_{i \in K}$ is not the set cover of S, then $\exists t \in S, t \notin \cup_{i \in K} S_i$. For each i, suppose $t \in f(v^i_{a_x})$, then let $g(v_i) = v^i_{a_x}$,by the previous theorem, g is a subgraph isomorphism and g is different from SI. □

The above theorem has immediate corollary:

Corollary 1. *The minimum set cover problem is reducible to solving the minimum solution verification set problem.*

This demonstrates that the decision variant of the minimum solution verification problem (finding a verification set with fewer than N queries) is NP-hard. If we note that any such set is a certificate for the problem, it follows that the problem is in NP, and so we have the following corollary.

Corollary 2. *The decision variant of the solution verification problem is NP-complete.*

4.2.2 Solving the Minimal Solution Verification Set Problem

In spite of the fact that determining if there is a solution verification set of a given size is NP-complete, it may still be possible to solve this problem if the solution space $\mathcal{F} := F(G_t, G_w)$ can be computed and is of a manageable size.

Given $\mathcal{F}$ and a solution $f \in \mathcal{F}$, we can reduce finding a minimal verification set to the minimum set cover problem in the following manner. For each subset of template vertices $A \subset V_t$, we define $\mathcal{F}_A := \{g \in \mathcal{F} : \exists t \in A, g(t) \neq f(t)\}$, the set of SIs that we can rule out if we know $f(t)$ for all $t \in A$. A solution verification set is any set $A \subset V_t$ for which $\mathcal{F}_A = \mathcal{F} \setminus \{f\}$. We then define $|V_t|$ sets, $S_1 = \mathcal{F}_{t_1}, \ldots, S_{|V_t|} = \mathcal{F}_{t_{|V_t|}}$. Our goal is then to determine an index set $I \subset \{1, 2, \ldots, |V_t|\}$ of minimal cardinality such that $\bigcup_{i \in I} S_i = \mathcal{F} \setminus \{f\}$. This is precisely the minimum set cover problem as defined in Definition 19.

We can solve the minimum set cover using a binary program where we introduce binary variables $z_1, \ldots, z_k \in \{0, 1\}$ where z_i represents whether or not we are using subset S_i in our set cover. We define a matrix $A \in \{0, 1\}^{k \times m}$ with entries a_{ij} for $i = 1, \ldots, k$ and $j = 1, \ldots, m$ where $a_{ij} = 1$ if element j is in subset S_i and 0 otherwise. We can then write the optimization problem we wish to solve with the following formulation:

$$\min_z \sum_{i=1}^{k} z_i \tag{4.2}$$

$$\text{s.t.} \sum_{i=1}^{k} a_{ij} z_i \geq 1, \qquad j = 1, \ldots, m \tag{4.3}$$

$$z_i \in \{0, 1\}, \qquad i = 1, \ldots, k \tag{4.4}$$

where the objective that we are minimizing in (4.2) is exactly the count of how many subsets we use. The constraints in (4.3) verify that each element in S is in at least one chosen subset. In practice, the matrix A often has a significant number of duplicate columns which we can safely drop without changing the solution set of the problem. We can then solve this integer program using any standard integer program solver; for this thesis, we use the state-of-the-art solver Gurobi [31]. Once we have solved this problem, the size of the given solution verifying set can then be used as a lower bound on the number of queries under any given template query strategy.

4.3 Querying strategies for template nodes

In this section, we will introduce various query strategies for the active learning problem in subgraph matching.

4.3.1 Local template centrality based strategies

The first strategies we consider are inspired by the assumption that querying a template node will reduce the candidate set on neighboring nodes more than other nodes. As a result, the strategies are based on the local regions containing neighbors of template nodes and prefers nodes with higher degree.

In [30], these strategies are shown to be effective to reduce the solution space with a limited number of queries.

- Maximum degree: choose the template node with the highest degree.

$$t = \arg\max_{t \in V_t} \deg(t) \tag{4.5}$$

- Max sum of candidates: choose the node with the largest sum of the number of candidates for neighboring template nodes and itself.

$$t = \arg\max_{t \in V_t} \left(|C(t)| + \sum_{t' \in N(t)} |C(t')| \right) \tag{4.6}$$

- Edge entropy: Let $t \in V_t$ and $t' \in N(t)$. Then let $N_{t'}^{tw}$ be the number of candidates for t' if t is matched to $w \in C(t)$. Let $N_{t'}^{t} = \sum_{w \in C(t)} N_{t'}^{tw}$. Then we can interpret $P_{t'}^{EE}(t = w) := N_{t'}^{tw}/N_{t'}^{t}$ as an estimate for the probability that t is mapped to w based on the possible mappings of the edge with endpoints t' and t. The edge entropy formula is then given as follows:

$$EE(v) = \sum_{t' \in N(t)} \sum_{w \in C(t)} -P_{t'}^{EE}(t = w) \log P_{t'}^{EE}(t = w) \tag{4.7}$$

4.3.2 Probabilistic Query Strategies

In this section, we introduce strategies for querying which attempt to estimate the probability with which a template vertex is assigned to a given world vertex. There are a variety of ways of establishing a probability space on the set of subgraph isomorphisms but the simplest involves granting all subgraph isomorphisms for a given template graph and world graph equal probability.

With this established, the probability that a template vertex $t \in V_t$ and world vertex $w \in V_w$ are paired together is simply the proportion of subgraph isomorphisms where they are matched:

$$P_{SI}(t = w) = \frac{|\{f \in F(G_t, G_w) : f(t) = w\}|}{|F(G_t, G_w)|}. \tag{4.8}$$

If we know certain template vertices are already assigned to world vertices, we can adjust the above definition to restrict $F(G_t, G_w)$ to those with the known assignments.

Directly computing all subgraph isomorphisms is in many cases computationally intractable owing to the NP-complete nature of simply finding one. Instead of fully enumerating all subgraph isomorphisms, we can instead try to approximate the above quantity based on local structures.

One such structure is the immediate neighborhood. We can attain an approximation for the number of solutions by asking how many mappings there are from the neighbors of a template vertex t to the neighbors of a world vertex w. Put concretely, we can define the set of **local subgraph isomorphisms** (LSI) for any template vertex, world vertex pair (t, w) in the following manner:

$$\begin{aligned} LSI(G_t, G_w, t, w, C) = \{f : N(t) \to N(w) : f \text{ injective}, \\ f(t') \in C(t'), \forall t' \in N(t)\}. \end{aligned} \tag{4.9}$$

This is precisely the set of subgraph isomorphisms from the neighborhood of t to the neighborhood of w with the additional requirement that each template vertex has candidates inherited from the original subgraph isomorphism problem.

We can then define an associated probability:

$$P_{LSI}(t = w) = \frac{|LSI(G_t, G_w, t, w, C)}{\sum_{w' \in C(t)|} |LSI(G_t, G_w, t, w', C)|}. \tag{4.10}$$

Computing the number of LSIs corresponds to counting mappings between $N(t)$ and $N(w)$ and ensuring that each template node is assigned a different world node. In the constraint programming framework, this is referred to as an *alldifferent* problem. We can reduce this problem to that of finding the permanent of a 0-1 matrix which lies in the P#-complete complexity class [82] suggesting that even this problem may be difficult to solve efficiently. In practice, the problem sizes are often small enough and we can exploit symmetry in the candidate sets to solve these relatively quickly.

In the case that this problem is still too costly to solve, we can compute a crude approximation to the number of LSIs by removing the injectivity requirement in which case we are counting the number of **local edge preserving mappings** (LEPM). This quantity is very easy to count as it is just given by the product of the sizes of each of the candidate sets for the all neighbors:

$$|LEPM(G_t, G_w, t, w, C)| = \prod_{t' \in N(t)} |C(t') \cap N(w)|. \tag{4.11}$$

Then our probability is defined in the same manner as before:

$$P_{LEPM}(t = w) = \frac{|LEPM(G_t, G_w, t, w, C)}{\sum_{w' \in C(t)|} |LEPM(G_t, G_w, t, w, C)}|. \tag{4.12}$$

We can also obtain a rough approximation of the number of SIs by instead considering EPMs for a spanning tree of the template. As we will discuss in Section 4.4, it is tractable to compute this and we present an algorithm for doing so. We denote the number of EPMs between a spanning tree T of template G_t world G_w where t is mapped to w by $STEPM(G_t, G_w, t, w, T)$. We then have an associated probability computed in the same manner:

$$P_{STEPM}(t = w) = \frac{|STEPM(G_t, G_w, t, w, T)|}{\sum_{w' \in C(t)} |STEPM(G_t, G_w, t, w', T)|}. \tag{4.13}$$

Once we have computed the associated probabilities, we can devise a variety of strategies for selecting query vertices. There are well-established strategies in active learning (surveyed here [71]) and three popular choices for determining queries are minimum confidence sampling, margin sampling, and maximum entropy sampling. We define these as follows:

- *Minimum Confidence Sampling*: Select the template vertex which is least confident about its most likely assignment.

$$t = \arg\min_{t \in V_T} \max_{w \in C(t)} P(t = w). \tag{4.14}$$

- *Margin Sampling*: Select the template vertex whose two most probable assignments are closest together in probability. If $w^* = \arg\max_{w \in C(t)} P(t = w)$, then this is given by

$$t = \arg\min_{t \in V_T} \left(P(t = w^*) - \max_{w \in C(t), w \neq w^*} P(t = w) \right). \tag{4.15}$$

- *Maximum Entropy Sampling*: Select the template vertex which maximizes entropy.

$$t = \arg\max_{t \in V_T} - \sum_{w \in C(t)} P(t = w) \log P(t = w). \tag{4.16}$$

Results from preliminary experiments from the `biochemical_reactions` dataset (to be introduced in full detail in Section 4.5) are depicted in Figure 4.2. In these experiments, we compare methods by the average amount of queries needed to uniquely determine a solution

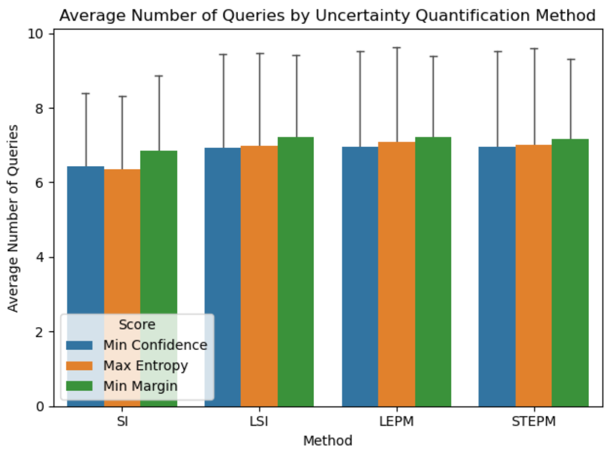

Figure 4.2: Average Number of queries needed to determine a solution to the subgraph matching problem on the `biochemical_reactions` dataset based on the approximation used for the number of SIs and the uncertainty quantification method. Error bars depict the standard deviation in the number of queries.

for a random selection of subgraph matching problems from this dataset. We observe that the maximum entropy and minimum confidence methods have similar performances with the minimum margin method doing worse for each method for computing probability. As these methods have negligible differences on average, we will primarily focus on the maximum entropy approach in this chapter.

4.3.3 Symmetry in Active Learning

Another approach for developing a query strategy involves analyzing the symmetry apparent in the subgraph matching problem. Symmetry is well-known for confounding general combinatorial problems by dramatically expanding the solution space. Algorithms which exploit symmetry [6, 66, 33, 61] can significantly reduce search time (sometimes by an exponential factor for highly symmetric problems). These algorithms generally identify nodes which are effectively interchangeable in that exchanging them in a subgraph isomorphism will produce another valid isomorphism.

There are various notions of symmetry which can be discussed in the context of the subgraph matching problem. The simplest is **structural equivalence** (considered in [66, 33, 61]) and two nodes are deemed structurally equivalent when they have the exact same set of neighbors (excluding each other). These nodes can be interchanged in any isomorphism and the new mapping will also be an isomorphism. A more complicated notion which includes structural equivalence is **automorphic equivalence**. Two nodes are automorphically equivalent if there is an automorphism $\phi : V \to V$ on the graph mapping one node to the other. Then given this automorphism ϕ and any subgraph isomorphism f, we can construct a new subgraph isomorphism $f \circ \phi$.

In the context of the active learning problem, symmetry can be an important factor to consider when deciding which nodes to query. If we have a group of M structurally equivalent template nodes, then for any subgraph isomorphism f, we can construct $M! - 1$ additional isomorphisms by simply permuting the images of these nodes. All of these isomorphisms are valid based on the information apparent in the problem. To discern which permutation is the true solution necessarily requires querying at least $M - 1$ of these nodes (the last may be determined by process of elimination).

This discussion naturally leads to the following proposition:

Proposition 1. *Suppose we have a satisfiable subgraph isomorphism problem with template* $G_t = (V_t, E_t)$ *and world* $G_w = (V_w, E_w)$*. Let* $V_t = \bigcup_{i=1}^{N} S_i$ *partition the template vertices into structural equivalence classes. Then at least* $\sum_{i=1}^{N}(|S_i| - 1)$ *template queries are needed to identify a unique solution.*

Discerning between isomorphisms which are generated by applying an automorphism to the template graph is more challenging. Relevant to this task is the notion of a base of the automorphism group of a template graph. A **base** of an automorphism group is a sequence of vertices $(v_1, v_2, \ldots, v_n)$ such that for any pair of automorphisms $\phi_1 \neq \phi_2$, the two sequences $(\phi_1(v_1), \phi_1(v_2), \ldots, \phi_1(v_n))$ and $(\phi_2(v_1), \phi_2(v_2), \ldots, \phi_2(v_n))$ are distinct. This definition implies that the values of ϕ on $v_1, \ldots, v_n$ uniquely identify ϕ. Hence, to distinguish isomorphisms generated by the automorphism group would involve querying all vertices of a base of the automorphism group.

The problem of generating a base for the automorphism group is difficult but the library `nauty` [56] in the process of finding generators for the automorphism group of a general graph will also find a base. Querying nodes which constitute a base for the automorphism group then may be a useful strategy for active learning which exploits symmetry.

4.4 Spanning-Tree-Based Estimation of the Subgraph Isomorphism Count

In this section, we develop a new method that is computationally tractable on larger multi-channel networks, especially those with long paths. The idea is motivated by the estimation of the number of subgraph isomorphisms in the CFL-Match algorithm [6]. The authors of this algorithm proposed a new data structure called the compact path index (CPI). Using our notation, the structure of CPI is defined by the rule: There is an edge between $v \in C(u)$ and $v_0 \in C(u_0)$ for adjacent nodes u and u_0 in CPI if and only if $E(v, v_0) = 1$ in G.

Here we use this data structure to estimate the frequency of solutions when the template is a path graph, and extended the method to the case when the template is a tree. In such a case, the estimation is the number of edge-preserving mappings generated by the current candidate list. For a more general template, we use a breadth first search (BFS) starting from the selected node to generate a spanning tree of the template, and do the estimation using the spanning tree.

4.4.1 Solution Frequency Estimation Algorithm

Solving the subgraph isomorphism problem is hard, even when the template structure is simple. This is in part because we need to solve an all-different problem due to the injectivity requirement on the nodes, and the counting of all assignments under all-different constraint is P# complete. Research has been done to estimate the cardinality of the solution space, [86][63]. However, these methods cannot be easily adapted to estimate the number of mappings that each world node participates in in the solution space due to the size of the world graph. A good estimation of this number is needed to inform the active learning problem. We use the number of possible edge-preserving mappings to the current candidate set to approximate the number of subgraph isomorphisms. By removing the injectivity constraint, we no longer need to solve the all-different problem, and complete the estimation in polynomial time.

4.4.1.1 Estimation for Path Graphs

We start from the estimation on the root of a template which is a path graph. First, we give the definition of a path graph for multichannel networks:

Definition 22 (Path graph)**.** *T is called a* ***path graph*** *if there exists an indexing of template nodes $V_t = \{x_1, x_2, \ldots, x_k\}$, such that the edge set of the template satisfies the following conditions:*

$$E_t(x_i, x_j) > 0 \Longleftrightarrow (x_i, x_j) \in \{(x_1, x_2), \ldots, (x_{k-1}, x_k)\}$$

Here, x_k is defined as the root of this path graph.

From the definition, if x_k is the root of the path graph template, then the template nodes can be indexed by $x_1, x_2, \ldots, x_k$ such that edges only exist in nodes with consecutive indices.

Definition 23. *The solution frequency estimation problem is defined as followed: Suppose x_k is a given node of a template G_t. Suppose $C(x_i)[j]$ represents the jth element in the candidate* *set $C(x_i)$. The solution frequency estimation problem is to approximate the cardinality of the following sets of subgraph isomorphism solutions f: $\{f \in F(G_t, G_w) | f(x_k) = C(x_k)[j]\}$ for $j = 1, 2, \ldots, |C(x_k)|$.*

If the template is a path graph with a root x_k, the solution frequency estimation problem with given node x_k is called the path graph root estimation problem.

Definition 24. *Under the setting of the path graph root estimation problem, we suppose the template is a path graph. the CPI adjacency matrix $M_{i,i+1}$ is a $0-1$ matrix of size $|C(x_i)| \times |C(x_{i+1})|$ defined as followed:*
if $E_w(C(x_i)[a], C(x_{i+1})[b]) > 0$, then $M_{i,i+1}[a,b] = 1$. Otherwise, $M_{i,i+1}[a,b] = 0$.

Under the setting of the path graph root estimation problem, the CPI graph is defined as follows:

Definition 25. *The CPI graph is a single-channel graph with $\sum_{i=1}^{k} |C(x_i)|$ nodes. The nodes are indexed by the following set: $V = \{(x_i, C(x_i)[j]), i = 1, 2, \ldots, k; j = 1, 2, \ldots, |C(x_i)|\}$. This means that each node corresponds to a unique template-candidate pair. The edge set is defined to be:*

$$\begin{aligned} &E((x_i, C(x_i)[a]), (x_j, C(x_j[b])) = 1 \Leftrightarrow |j - i| = 1 \\ &\textit{and} \;\; E_w(C(x_i)[a], C(x_j)[b]) \geq E_t(x_i, x_j). \end{aligned}$$

Otherwise,

$$E((x_i, C(x_i)[a]), (x_j, C(x_j[b])) = 0$$

This means the edges of CPI graph contains all the template-candidate pairs such that the template node and candidate are simultaneously neighboring to each other.

The following theorem gives a formula to calculate the exact number of edge preserving mappings for path graph templates.

Theorem 26. *In the setting of the path graph template root estimation problem, the cardinality of edge-preserving mappings* $|\{g \in EPM | g(x_k) = C(x_k)[j]\}| = (\mathbf{1}^T \cdot \Pi_{i=1}^{k-1} M_{i,i+1})[j]$.

Proof. We first prove the following mapping m is a bijection between paths of length k in the CPI graph and edge-preserving mappings: $m(g) = \{(x_i, g(x_i)), i = 1, 2, \ldots, k\}$, by the definition of CPI graph, the $\{(i, g(x_i)), i = 1, 2, \ldots, k\}$ is a path in CPI graph. By the definition of g, m is injection. On the other hand, for all the node lists $\{(i, g(x_i)), i = 1, 2, \ldots, k\}$ that is a path in CPI graph, by definition, m^{-1} is also an edge preserving mapping and injection. So m is bijection.

Suppose the adjacency matrix of the CPI graph is denoted by ADJ; it is a matrix of size $\sum_{i=1}^{k} |C(x_i)|$. We use the notation $ADJ(C(x_a), C(x_b))$ to represent the block of ADJ indexed by the following nodes:

$\{(x_a, C(x_a)[j]), j = 1, 2, \ldots, |C(x_a)|\}$ and $\{(x_b, C(x_b)[j]), j = 1, 2, \ldots, |C(x_b)|\}$ Then $ADJ(C(x_i), C(x_i + 1)) = M_{i,i+1}$,$ADJ(C(x_i + 1), C(x_i)) = M_{i,i+1}^T$, and all other blocks are 0 matrices.

By induction, $ADJ^k(C(x_1), C(x_k)) = \Pi_{i=1}^{k-1} M_{i,i+1}$. So, for any $s \in \{1, 2, \ldots, |C(x_1)|\}$,$j \in \{1, 2, \ldots, |C(x_k)|\}$, the number of paths of length k from $(x_1, C(x_1)[s])\}$ to $(x_k, C(x_k)[j])$ is given by $ADJ^k(C(x_1), C(x_k))[s, j] = \Pi_{i=1}^{k-1} M_{i,i+1}[s, j]$. By taking the sum over s, we get the number of edge preserving mappings $(\mathbf{1}^T \cdot \Pi_{i=1}^{k-1} M_{i,i+1})[j]$. This completes the proof. □

4.4.1.2 Estimation for Trees

A tree in graph theory is defined for single channel graphs as a connected acyclic undirected graph. A tree is also a notion in data structure usually visualized by a tree in the graph theory sense, and stores the connection and hierarchy of the nodes. These two definitions are linked together by assigning a root to the tree.

The following definitions are aimed to extend the definition of graphical tree to multichannel graphs and define the tree data structure similarly by assigning a root node.

Definition 27 (Underlying graph)**.** *Given a directed or undirected multichannel graph G, the* ***underlying graph*** *G^u is a single channel, undirected simple graph which is defined as followed:*

$$V_G = V_{G^u},$$
$$(\mathbf{x}, \mathbf{y}) \in E_{G^u} \Longleftrightarrow E_G(x,y) > 0 \text{ or } E_G(y,x) > 0.$$

Definition 28. *A multichannel graph G has a tree structure if and only if the underlying graph G^u is a tree.*

The definition of rooted tree is generalized from [5] is equivalent to the tree data structure.

Definition 29 (Rooted tree)**.** *Suppose G is a tree or G is a multichannel graph with tree structure, and $u \in V_G$, then the pair (G, u) is called a rooted tree with root u.*

Definition 30 (Parent, Ancestor, Child, Descendant, and Sibling)**.** *For a rooted tree(T, r), suppose w is any node other than r. If G is a tree, then there is a unique path P in G that connects w and r. If G has tree structure, then there is a unique path P in G^u that connects w and r. Suppose $P = (v_0, v_1, \ldots, v_k)$, where $v_0 = r, v_k = w$. Then v_{k-1} is called the parent of w, w is called the child of v_{k-1}. $v_0, v_1, \ldots, v_{k-1}$ are called the ancestors of w, and w is the descendant of $v_0, v_1, \ldots, v_{k-1}$. Vertices with same parent are called siblings to each other.*

Definition 31 (Subtree for rooted tree)**.** *Suppose (T, r) is a rooted tree, and $w \in V_T$. The subgraph T_w generated by w and all its descendants also has a tree structure. (T_w, w) is called the subtree of (T, r) with root w.*

We call the formula given in Theorem 4.4.1.1 *pathEPM*, and use the following divide and conquer algorithm which we call $TreeEPM$ to calculate the number of EPMs. To avoid repeat calculations, we pre-calculate all the CPI matrices and store them in a hash map M. We denote by $M(x, y)$ the CPI matrix for template nodes x and y.

Algorithm 5 TreeEPM

```
Input: Template $T$ with tree structure, root $x$, CPI matrix dictionary $M$
$y_1 \dots, y_m \leftarrow Children(x)$
$(T_1, y1), \dots, (T_m, y_m) \leftarrow Subtrees$
if $T$ is a path graph then:
    Return $pathEPM(T, x, M)$
else
    for $T_i \in Subtree(T)$ do
        $subepm_i \leftarrow TreeEPM(T_i, y_i, M) * M(y_i, x)$
    Return $subepm_1 \odot subepm_2 \odot \dots subepm_m$
```

In the above algorithm, $\odot$ denotes the element-wise product of two vectors. This algorithm returns the exact number of EPMs regarding each candidate node of the selected root x.

Theorem 32. *The cardinality of edge-preserving mappings* $|\{g \in EPM(T)|g(x) = C(x)[j]\}| = TreeEPM(T, x, M)[j]$.

Proof. For the rooted tree (T, x), suppose x has children $y_1, y_2, \dots, y_m$, the subtrees rooted at $y_1, y_2, \dots, y_m$ are $(T_1, y1), (T_2, y2), \dots, (T_m, y_m)$ We proceed by induction on the number of the nodes of the tree T. For the base case where $|V_T| = 1$, then T is a path graph, $pathEPM(T, x, M)$ will return the correct result. Now suppose when $|V_T| < k$, the proposition is true, then for $|V_T| = k$, T_i are subtrees of size less or equal than k. $treeEPM(T_i, y_i, M)$ will give correct result.That is to say $|\{g \in EPM(T_i)|g(y_i) = C(y_i)[j]\}| = TreeEPM(T_i, y_i, M)[j]$. Since $|\{g \in EPM(T_i \cup x)|g(x) = C(x)[j]\}| = \sum_{r=1}^{C(y_i} |\{g \in EPM(T_i \cup x)|g(y_i) = C(y_i)[r], g(x) =$

$C(x)[j]\}|$,we have

$$
\begin{aligned}
&|\{g \in EPM(T_i \cup x) | g(x) = C(x)[j]\}| \\
&= \sum_{r=1}^{C(y_i)} |\{g \in EPM(T_i) | g(y_i) = C(y_i)[r]\}| \\
&\quad \times |\{g \in EPM(y_i, x) | g(y_i) = C(y_i)[r], g(x) = C(x)[j]\}| \\
&= \sum_{r=1}^{C(y_i)} TreeEPM(T_i, y_i, M)[r] * M(y_i, x)[r, j] \\
&= TreeEPM(T_i, y_i, M) * M(y_i, x)[j]
\end{aligned}
$$

Since $\{g \in EPM(T) | g(x) = C(x)[j]\}$ is the Cartesian product of the sets $\{g \in EPM(T_i \cup \{x\}) | g(x) = C(x)[j]\}$, we can conclude the theorem is true when the $|V_t| = k$.

□

4.4.1.3 Extension to general templates

For a general template, we do not have a tree structure naturally generated by the template. In this setting, we use a breadth first search to produce a spanning tree of our template. We then can estimate the number of EPMs on our general template by using the algorithm in the previous section on the spanning tree. Then we give the following estimation algorithm for a general template. For a selected node x, we use the TreeEPM algorithm on the rooted tree structure with root x. The algorithm returns a vector of length $|C(x)|$ recording an estimate of the number of EPMs that map x to each element in $C(x)$. In Figure 4.3, we show the example of a template and its related data structures in each step of the EPM estimation.

4.4.2 Complexity analysis of Spanning Tree Entropy method

We introduce first some terminology to be used in the complexity analysis. First, we denote by $|E_t|$, $|E_w|$, $|E^u_w|$, $|E^u_t|$ the number of edges in the template, world, and underlying graphs

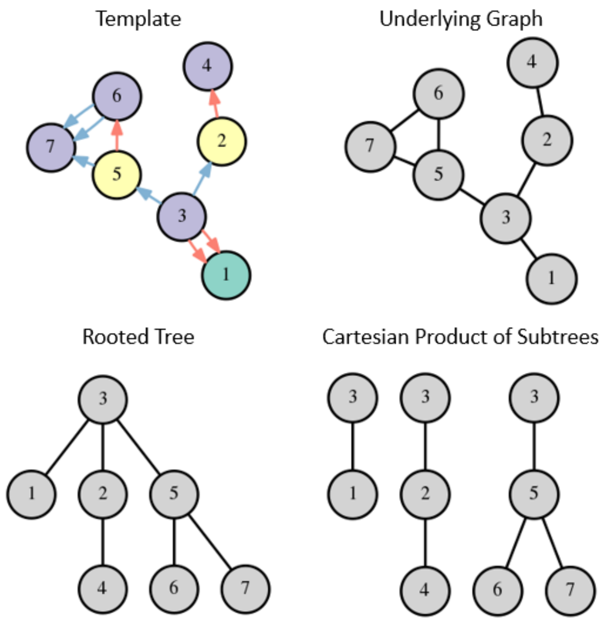

Figure 4.3: The figure above shows a toy example of estimating the edge preserve mappings for the candidate of the selected node with label 3. The underlying graph of the template is shown on the top right. By using the BFS starting from node 3, we can get a rooted tree with parent/children relations shown in the bottom left. By looking at the subtrees, the estimation can be calculated by the Cartesian product of the estimation from three subproblems with smaller size.

of template and world respectively. We denote by $C^{\max}$ the maximal size of a candidate set $C(x)$. We denote the maximal degree of the template by $D_t^{\max}$ and the average degree of the underlying graph of the world D_u^{avg}.

There are three steps for the estimation algorithm, the calculation of CPI matrix hash map M, the multiplication of matrices in the path, and the Cartesian product constructed in the tree structure. First, we give an upper bound for the worst case complexity. We need to compute a CPI matrix for all the adjacent pairs of nodes in the template. So the numbers of CPI matrices to compute is equal to N_{eut} and the size of each CPI matrix is $C^{\max} \times C^{\max}$. So the complexity of this part is bounded by $O(|E_t^u|(C^{\max})^2)$. Suppose we already selected a node x, to construct the tree, we need to visit all the nodes in the template once. Therefore, the complexity of this part is bounded by $O(|V_t|)$

Then, we need to perform the multiplication of matrix for every edge in the BFS tree, and the number of edges is equal to $|V_t|-1$. For each node in BFS tree, the size of the Cartesian product is bounded by $|V_t|-1$, and the cost of computing each Cartesian product is bounded by $O(C^{\max})$. So the total cost of this step is bounded by $O((C^{\max})^3|V_t|) + O(C^{\max} * |V_t|) = O((C^{\max})^3|V_t|)$.

We then need to select all nodes and repeat the calculation. So the total time complexity of the algorithm is bounded by $O((C^{\max})^3|V_t|^2 + |E_t^u|((C^{\max})^2)$. That means the algorithm can be completed within polynomial time. However, this bound of time complexity can be huge if the number of candidates for any vertex is large.

In certain cases, the template and the world graph both are sparse. Then from the sparsity of template, we assume $|E_t^u| = O(|V_t|)$. From the sparsity of world graph, the expectation of number of nonzero elements of each column of the CPI matrix is equal to D_u^{avg}. Therefore, the complexity of matrix multiplications for a BFS tree will be $O((C^{\max})^2|V_t|D_u^{\text{avg}})$ instead of $O((C^{\max})^3|V_t|)$. Hence, the expectation of total complexity with the assumption of sparsity is $O((C^{\max})^2|V_t|^2 D_u^{\text{avg}})$.

4.5 Experiments on Single Channel Networks

In this section, we describe a series of experiments performed on a collection of real and synthetic single channel subgraph isomorphism problems. We consider ten different strategies for querying nodes which are listed as follows:

1. MC (Most Candidates): Node with the most candidates.
2. MD (Minimum Degree): Node with the lowest degree.
3. MNCS (Maximum Neighbor Candidate Sum): Node whose neighbors have the most candidates as given by (4.6).
4. ME (Maximum Entropy): Node given by (4.16) with P_{SI} defined as in (4.8).
5. MLE (Maximum Local Entropy): Node given by (4.16) with P_{LSI} defined as in (4.10).
6. MLE$\sim$ (Maximum Local Entropy Approximation): Node given by (4.16) with P_{LSH} defined as in (4.12).
7. R (Random): Random node.
8. EE (Edge Entropy): Node with the highest edge entropy as defined in (4.7).
9. STE (Spanning Tree Entropy): Node given by (4.16) with P_{STEPM} as defined in (4.13).
10. O (Optimal): The optimal node to query as given by solving (4.2).

In the event that multiple nodes tie on one of these criteria, out of these nodes, the node with minimal index which has not been queried yet will be selected.

For our study, we will consider the benchmark dataset for subgraph matching problems compiled by Solnon [74]. We describe briefly some of the datasets included in this problem suite. The *LV* dataset is composed of a variety of graphs with theoretically interesting properties (e.g., biconnected, triconnected, highly symmetric, etc.). The *SI* dataset has four

Table 4.1: Benchmark Dataset Statistics

		Template						World					
Dataset	#	# Nodes		# Edges		Density		# Nodes		# Edges		Density	
	Inst	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
LV-easy	26	10	435	10	2520	0.027	1.000	10	2000	45	2592	0.001	1.000
SI-easy	446	40	777	43	2047	0.006	0.201	200	1296	299	4377	0.004	0.098
bio-easy	739	9	68	8	90	0.036	0.417	9	386	12	886	0.012	0.423
images	10	5	11	6	13	0.236	0.600	4838	4838	7067	7067	0.001	0.001
LV-hard	246	10	280	10	1848	0.002	1.000	42	6671	114	209000	0.001	1.000
SI-hard	249	40	518	41	1669	0.006	0.197	200	1296	299	7788	0.004	0.191
bio-hard	554	9	184	8	355	0.021	0.423	25	386	44	886	0.012	0.160
phase	47	30	30	128	387	0.294	0.890	150	150	4312	8740	0.386	0.782

different graph matching problem sets involving bounded valence graphs, modified bounded valence graphs, meshes, and Erdős–Rényi graphs. There are two sets of instances representing images for pattern recognition problems, *images-cv* and *images-pr* [18, 75] which we combine into one dataset *images*. The *biochemical_reactions* dataset [29] represents matching problems on systems of biochemical reactions. The last dataset under consideration is the *phase* dataset [53] are instances involving randomly generated Erdős–Rényi graphs known to be very difficult.

For our experiments, we only consider problems for which there are at least ten subgraph isomorphisms. From these, we divide our datasets into "easy" and "hard" instances. Easy instances are those for which we can fully enumerate the solution space and therefore can compute the optimal amount of queries as in Section 4.2. Hard instances are the subgraph matching problems where we do not have the full solution space and as such we cannot compute the number of queries using the O or ME methods. We also include in this set, problems for which the MLE does not find terminate within ten minutes. A compilation of basic graph statistics for these datasets is presented in Table 4.1.

For these problems, we select ten random solutions from the solution space for a given subgraph matching problem. For the easy instances, this is uniformly chosen over all solutions, and for hard instances, it is randomly chosen from a selection of solutions by the Glasgow solver [54] within one minute. Then for each solution, we proceed using the framework described in Algorithms 4. We count the number of queries made and then record the average number of queries required for a given subgraph isomorphism problem for each querying strategy.

The results for the easy data sets are presented in Figure 4.4. We also consider the average percent gap between a given method's number of queries and the best method for a specific problem. We observe that the ME method is nearly optimal for the *biochemical_reactions* and *SI* datasets and requires 20% more guesses on the other datasets. The next best methods are MLE, MLE∼, and STE which all approximate the ME method and require between 15 and 40% more guesses across the datasets. Empirically, there does not seem to be much difference in using MLE or the approximation MLE∼ with MLE∼ even outperforming MLE on certain datasets. The EE and MNCS methods on average perform worse than picking vertices at random. From the disparity between the optimal and the other methods, there remains significant work to be done to develop computationally tractable methods which can close this gap.

The average number of queries for the hard instances is presented in Figure 4.5 along with the average percent additional queries needed over the best method.As we do not have the full solution set computed, we cannot consider the O or ME methods, and we also exclude the MLE method due to its long computation time. These problems require significantly more queries, and we observe less variation in the number of queries across methods. On average, we observe that the MLE∼ and STE methods perform the best with MC not too far behind. The random method on average requires 10-15% more queries over the best of these methods typically.

As discussed in Section 4.3.3, graph symmetry plays a significant role in subgraph match-

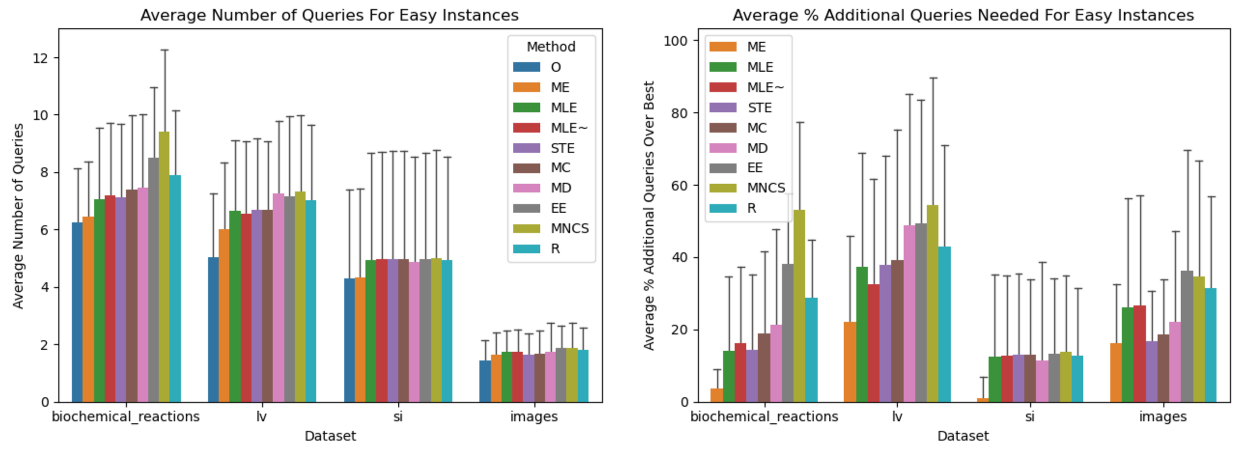

Figure 4.4: Bar charts depicting (left) the average number of queries and (2) average percent additional queries over the best query strategy for the easy problems in Table 4.1. Error bars depict the standard deviation. The methods listed are described in Section 4.5.

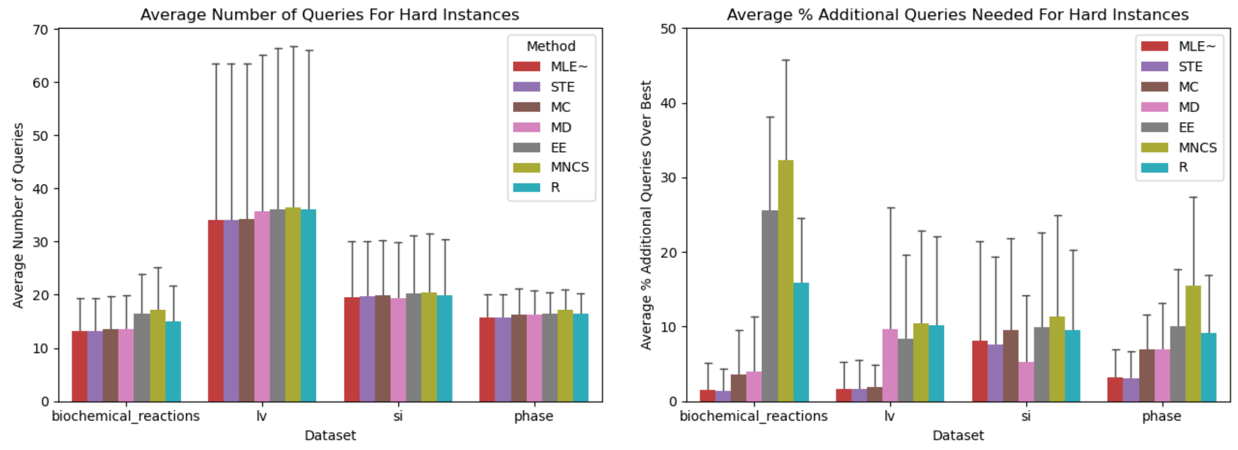

Figure 4.5: Bar charts depicting (left) the average number of queries and (right) average percent additional queries over the best query strategy for the hard problems in Table 4.1. Error bars depict the standard deviation. The methods listed are described in Section 4.5.

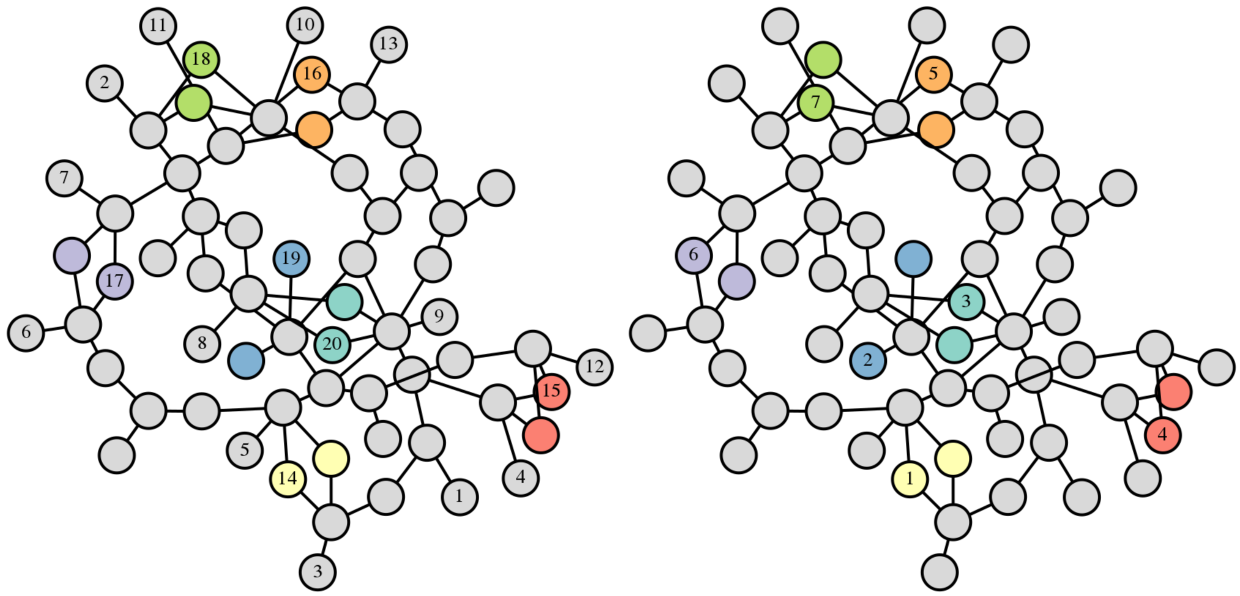

Figure 4.6: Two separate query strategies applied to an example template graph from the biochemical reactions dataset. The numbered nodes indicate the order in which template nodes are queried. On the left, the MLE query strategy is used, and on the right the ME query strategy is used. Non-gray nodes of the same color are structurally equivalent.

ing and neglecting it may confound some of the methods presented in this section. The biochemical reactions data set is a highly symmetric data set, and as a result of Proposition 1 necessarily require a high number of queries to distinguish a solution. For some query strategies, they systematically avoid the structurally equivalent nodes leading to high query counts. We observe this behavior in Figure 4.6 where the template graph from the biochemical reactions dataset has multiple sets of structurally equivalent vertices. The ME strategy which picks out the structurally equivalent vertices to query first finds the ground truth solution after only seven queries whereas the MLE strategy requires 20 queries. The reason for the significantly higher count comes from how the MLE approach leaves these structurally equivalent vertices for the end as other vertices have higher local entropy values at the stage in which they are queried.

In Figure 4.7, we compare two equivalence-based improvements on the MC strategy to

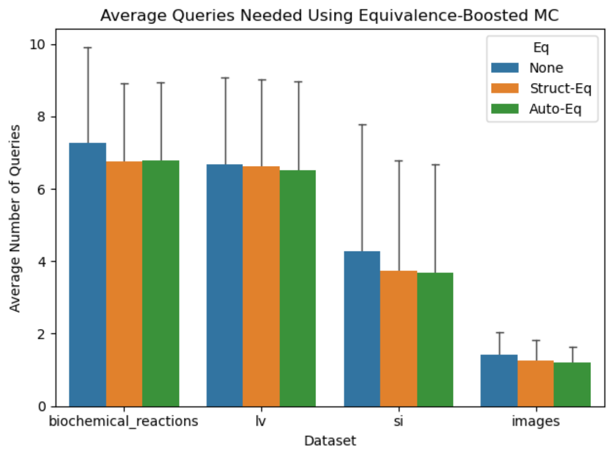

Figure 4.7: Average number of queries made before a solution is found on various single channel datasets when using the MC method with equivalence-informed queries.

the base MC approach. As discussed in Section 4.3.3, the structural equivalence approach queries all but one member of each structural equivalence class before considering other template vertices. The automorphic equivalence approach first queries template vertices which constitute a base of the automorphism group. As can be seen in the figure, these equivalence-informed improvements on the base methods can make significant improvements on the average number of queries required to determine a solution. On average, we can save nearly half a query for particular symmetric datasets like the *biochemical_reactions* and *SI* graphs, and on particular examples as in Figure 4.6, using symmetry is essential.

4.6 Experiments on Multichannel Networks

In this section, we discuss the performance of algorithms on multichannel subgraph isomorphism problems. The real world data include a transportation network in Great Britain [26] and COVID data [93]. The synthetically generated dataset, Ivysys v7 [2], was generated as part of the DARPA-MAA program.

4.6.1 Multichannel Datasets

The Great Britain Transportation Network [26] is comprised of the public transportation dataset available through the United Kingdom open-data program [81] with timetables of domestic flights in the UK. It is a multiplex time-dependent network. There are six channels involving different transportation methods, including bus, air, ferry, railway, metro, coach. This dataset has 262,377 nodes and 475,502 edges. The original dataset can be found at [27]. The authors of [59] have an online interactive map [35] for users to visualize the template.

The Ivysys data sets [2] concern interactions between individuals such as emails, phone calls, financial transactions. The template and world graphs are separately generated to match the degree distribution and email behavior of the Enron email dataset. The template graph of Ivysys v7 data has 92 nodes and 195 edges. The world graph has 2,488 nodes and 5,470,970 edges.

The COVID dataset studies the problem of extracting knowledge automatically from casual relations between biochemical entities. In [93], a knowledge graph is constructed using multiple sources, including the COVID-19 Open Research Dataset [83], the Blender Knowledge Graph [84], and the comparative toxigenomics database [20]. The authors of [93] then create a query representing how COVID 19 might cause a cytokine-storm in patients as well as other possible confounding factors. The template has 28 nodes and 38 edges. The world graph has 87,580 nodes and 1,736,985 edges.

The multichannel network data sets differ from the single channel data sets in terms of size of the networks and number of solutions. Previous research [87, 57] gives a lower bound of the number of solutions. To be specific, the lower bounds are $2.34e{+}15, 7.45e{+}21, 7.82e{+}103$ respectively for British Transportation, Ivysys v7 and Covid data. We present two ways to generate ground truth data for our experiment.

4.6.2 Methods for Generating Ground Truth Data

For the British Transportation Network, there is a small set of locations that interact with each other through all channels (excluding airlines, since this channel is very sparse). If a location involves all five non-air channels in the network, we assume that it is important. There are only three nodes that interact in the five non-air channels, and they randomly chose one of them as the template center, specifically the Blackfriars Station in London. Starting from this node, we do a random walk which terminates after it has visited 53 nodes. We repeat this process 100 times to produce 100 templates. These templates have the same number of nodes as the template used in [59]. In the process of creating the template, We record the names of stations visited by the random walk and use this as the ground truth of the subgraph isomorphism problem. This dataset is abbreviated as *BTN* in Table 4.2. We also generate the templates with 100 nodes in the same way. These templates are denoted by *BTN100* in Table 4.2.

For the *Covid* data and *Ivysysv7* data, the templates are generated from realistic problems instead of a random walk. However there are a large number of isomorphisms so we develop a randomized selection algorithm to choose one isomorphism as a ground truth to be "found" by our active learning method. This method iteratively assigns a random template node to its random candidate until we get a unique solution. We repeat this process 100 times to produce 100 ground truth isomorphisms.

4.6.3 Numerical Experiments

In our experiments, the templates in some of the datasets have unique structures. For example, as a result of random walk, the templates of *BTN* data usually contain several long paths. This is the same for the larger *BTN* templates with 100 nodes. But *BTN100* has longer random walk length, so the templates tend to form a core around the node representing starting station. The template of *Ivysysv7* data represents e-mail activity,

Algorithm 6 Random Selection of Ground Truth

```
Input: Template $G_t$, World $G_w$
$Match = \emptyset$
$C \leftarrow$ Filter($G_t$, $G_w$)                              ▷ Initialize domains
$Solutions = SolveSI(G_t, G_w, C, Match)$     ▷ Solve the SI problem with given matching
**while** $|Solutions| \neq 1$ **do**:
    **if** $|Solutions| > 1$ **then**:
        $u = randomselect(G_t), v = randomselect(C(u))$
        $Match = Match \cup (u, v)$
        $Solutions = SolveSI(G_t, G_w, C, Match)$
    **if** $|Solutions| = 0$ **then**:
        $Match = \emptyset$
        $Solutions = SolveSI(G_t, G_w, C, Match)$
Return $Solutions$
```

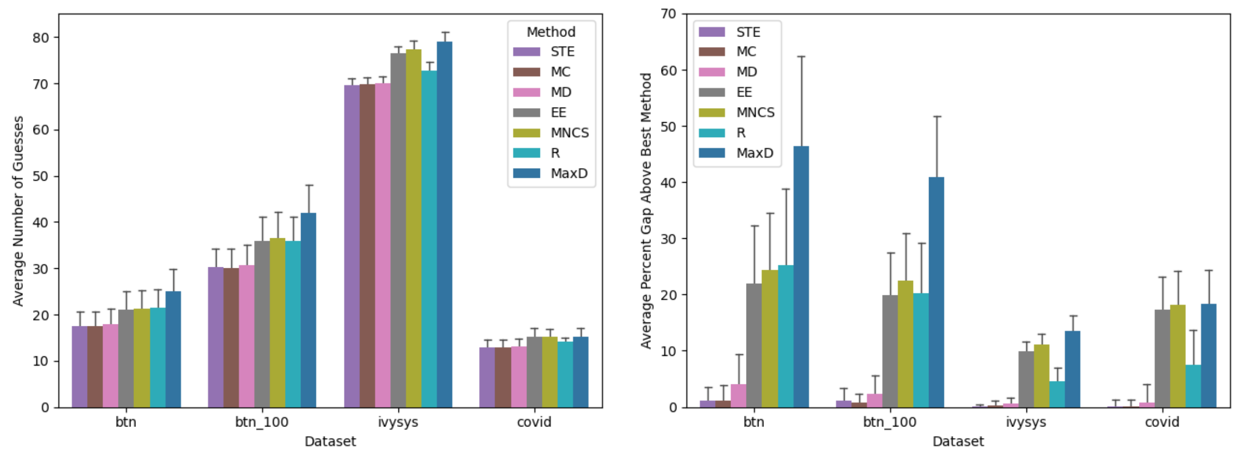

Figure 4.8: Bar charts depicting (left) the average number of template queries and (right) the average percent of additional queries over the best method for each problem in each multichannel data set. All methods are given in Section 4.5 except MaxD which represents the strategy of choosing the maximal degree node.

Dataset	$\lvert V_t \rvert$	$\lvert V_w \rvert$	Log of Search Space	MaxD	MNCS	STE	MD	R	EE	MC
BTN	53	262377	75.02	25.07	21.32	17.41	17.92	21.44	20.96	17.4
BTN100	100	262377	110.57	42.06	36.64	30.25	30.66	35.93	35.88	30.15
Ivysysv7	94	2488	262.11	78.96	77.32	69.64	69.98	72.73	76.46	69.81
Covid	28	87580	29.86	15.28	15.28	12.96	13.0	14.1	15.16	13.15

Table 4.2: This table shows the average of the statistics of different datasets and the average number of nodes to query for different active learning results. The search space of a dataset is defined to be the product of the size of candidate sets. We take the logarithm with base 10 and calculate the average. See Section 4.5 for definition of the methods.

and there are several components in the template that form star graphs. There are many structural equivalence classes in the template [87]. Table 4.2 shows some statistics related to datasets and the average numbers of queries needed for each active learning method. Figure 4.8 is a bar chart of the average number of queries and the average percent of additional queries compared to the best method.

The experimental results show that under our problem setting, assuming equal cost for querying each template node, less constrained nodes tend to be more important. By simply querying the node of least degree, we often get better results than most methods that need much more calculation.

The STE strategy gives similar query numbers as the minimum degree. But the way of selecting nodes is different. In our experiment, for a given ground truth, we include the nodes in the query list of any active learning method if querying all the other nodes can not reduce the solution space to one. In addition, for any non-trivial equivalent class of size k, we need to include at least $k-1$ nodes in the query list. We show a case study in figure 4.9 to illustrate how this works with a particular instance from the BTN dataset; the theoretical optimal query is reached by an exhaustive search on all the selections. The optimal query is one node less than both methods. Notice that the STE stategy queried an unnecessary

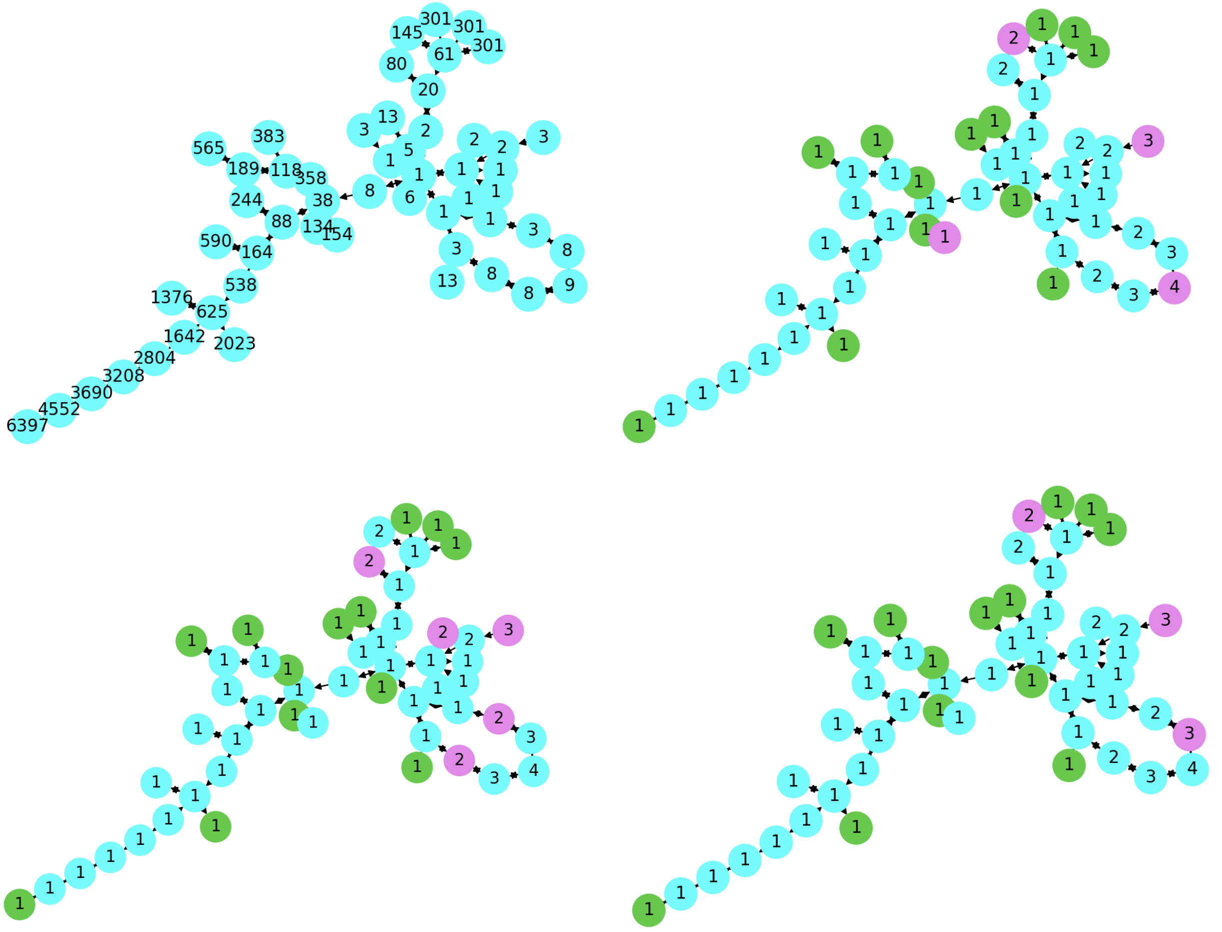

Figure 4.9: (Top left) Number of candidate of nodes for one of the British Transportation Network templates. In the remaining panels we show the selection of nodes using different active learning queries. Color codes are as follows: Green: the nodes that must be included. Purple: other nodes selected by the active learning method. The number in the circle represent the number of candidates after querying the green nodes. (Top right) EPM entropy method uses 17 queries. (Bottom Left) The minimum degree method uses 18 queries. (Bottom right) A theoretically optimal solution with 16 queries.

node (the node with 1 inside the circle). That node can be determined by the all-different constraint and the information from other nodes. minimum degree method does not differ between the nodes with same degree, so the performance of this method depends on some randomness.

4.7 Conclusions and Future Work

In this chapter, we present a rigorous mathematical treatment of the active learning query problem for subgraph matching where only one ground truth solution is desired. This has not been considered in the prior literature however it is an important problem for real-world problems in which a subgraph-matching problem may have a multitude of solutions yet the solution space can be greatly reduced with additional information brought by a subject matter expert. Such *active learning* problems are well-known in semi-supervised machine learning but have not been studied in the subgraph matching setting. We prove that finding the optimal template vertices to query to be NP-complete even if the solution space and the ground truth are known. We present several strategies for determining template queries, some based on simple graph structure and others which attempt to estimate the probability of matching template and world nodes. We design an inexpensive and fast method to estimate the solution space based on the spanning tree. We assess our results on benchmark data and compare the performance of different strategies on single channel and multichannel network datasets with varying graph size and number of solutions. Our experiments show that for single channel networks, the maximum entropy strategy which requires the calculation of the solution space is nearly optimal and other methods which approximate this approach are close but with much room for improvement. For large multichannel graphs, several methods have similar behavior, with the spanning tree entropy method performing best on average.

There are many open problems not discussed in this chapter. One obvious one is the consideration of more diverse active learning strategies. For example, the information of query could be related to world graph nodes rather than template graph. Likewise we do not consider combinations of different strategies in sequence but rather just consider iterations of a single strategy. Extensions on the subgraph isomorphism problem is another possible direction for future research. We can introduce noise to the problem and study the active learning for inexact subgraph matching problem [78, 43, 76]. Another class of problems are path-based queries where one desires to find a path from A to B in the network of e.g. shortest length, rather than finding an exact or inexact match of a template graph.

We also only considered sequential queries. Future work could examine batch active learning scenarios - especially in the case of large complex graphs or queries done on the world graph rather than the template.

4.8 Limitations

This work introduces a new set of problems in subgraph matching. We propose some strategies that have limitations. For example, the spanning tree based strategy assumes that the template graph is sparse with a tree-like structure. This strategy may be less informative with templates that are not sparse or that have a cyclical structure. We do not have an efficient strategy for finding the optimal solution to the active learning problem. We also note that the sequential strategies mentioned here have a total number of queries that are of the same order of magnitude as the number of template graph nodes. The example in fig. 12 shows an optimal choice that likewise has the same order of magnitude, indicating that for such combinatorially complex solution spaces one will need to provide additional information about a large fraction of the template nodes in order to reduce the solution space to one isomorphism. This last issue is a limitation of the problem rather than our suggested strategies.

REFERENCES

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[2] K. O. Babalola, O. B. Jennings, E. Urdiales, and J. A. DeBardelaben. Statistical methods for generating synthetic email data sets. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3986–3990, Dec 2018.

[3] Yunsheng Bai, Derek Xu, Yizhou Sun, and Wei Wang. Glsearch: Maximum common subgraph detection via learning to search. In *International Conference on Machine Learning*, pages 588–598. PMLR, 2021.

[4] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. Agnostic active learning. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 65–72, Pittsburgh, Pennsylvania, USA, June 2006. Association for Computing Machinery.

[5] Edward A Bender and S Gill Williamson. *Lists, decisions and graphs.* S. Gill Williamson, 2010.

[6] F. Bi, L. Chang, X. Lin, L. Qin, and W. Zhang. Efficient subgraph matching by postponing cartesian products. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1199–1214. ACM, 2016.

[7] Kenneth D Blaha. Minimum bases for permutation groups: the greedy approximation. *Journal of Algorithms*, 13(2):297–306, 1992.

[8] Wenbin Cai, Ya Zhang, and Jun Zhou. Maximizing Expected Model Change for Active Learning in Regression. In *2013 IEEE 13th International Conference on Data Mining*, pages 51–60, December 2013. ISSN: 2374-8486.

[9] Alessio Cardillo, Jesús Gómez-Gardenes, Massimiliano Zanin, Miguel Romance, David Papo, Francisco Del Pozo, and Stefano Boccaletti. Emergence of network features from multiplexity. *Scientific reports*, 3(1):1–6, 2013.

[10] V. Carletti, P. Foggia, A. Saggese, and M. Vento. Introducing VF3: A new algorithm for subgraph isomorphism. *Graph-Based Representations in Pattern Recognition*, pages 128–139, 2017.

[11] V. Carletti, P. Foggia, A. Saggese, and M. Vento. Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with vf3. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):804–818, April 2018.

[12] V. Carletti, P. Foggia, and M. Vento. Vf2 plus: An improved version of vf2 for biological graphs. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 168–177. Springer, 2015.

[13] F. Celli, F. M. L. Di Lascio, M. Magnani, B. Pacelli, and L. Rossi. Social Network Data and Practices: the case of Friendfeed. In *International Conference on Social Computing, Behavioral Modeling and Prediction*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010.

[14] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. *How and Why Pattern Recognition and Computer Vision Applications Use Graphs*, pages 85–135. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[15] Donatello Conte and Francesc Serratosa. Interactive online learning for graph matching using active strategies. *Knowledge-Based Systems*, 205:106275, 2020.

[16] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. on Pattern Analysis and Machine Intell.*, 26(10):1367–1372, Oct 2004.

[17] J. A. Cottam, S. Purohit, P. Mackey, and G. Chin. Multi-channel large network simulation including adversarial activity. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3947–3950, 10 2018.

[18] Guillaume Damiand, Christine Solnon, Colin De la Higuera, Jean-Christophe Janodet, and Émilie Samuel. Polynomial algorithms for subisomorphism of nd open combinatorial maps. *Computer Vision and Image Understanding*, 115(7):996–1010, 2011.

[19] Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 208–215, Helsinki, Finland, July 2008. Association for Computing Machinery.

[20] Allan Peter Davis, Thomas C Wiegers, Jolene Wiegers, Cynthia J Grondin, Robin J Johnson, Daniela Sciaky, and Carolyn J Mattingly. Ctd anatomy: analyzing chemical-induced phenotypes and exposures from an anatomical perspective, with implications for environmental health studies. *Current research in toxicology*, 2:128–139, 2021.

[21] Manlio De Domenico, Antonio Lima, Paul Mougel, and Mirco Musolesi. The anatomy of a scientific rumor. *Scientific reports*, 3(1):1–9, 2013.

[22] Adrie CM Dumay, Rob J van der Geest, Jan J Gerbrands, Eric Jansen, and Johan HC Reiber. Consistent inexact graph matching applied to labelling coronary segments in arteriograms. In *11th IAPR International Conference on Pattern Recognition. Vol. III. Conference C: Image, Speech and Signal Analysis,*, volume 1, pages 439–442. IEEE Computer Society, 1992.

[23] Frank Emmert-Streib, Matthias Dehmer, and Yongtang Shi. Fifty years of graph matching, network alignment and network comparison. *Information sciences*, 346:180–197, 2016.

[24] Pasquale Foggia, Gennaro Percannella, and Mario Vento. Graph matching and learning in pattern recognition in the last 10 years. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(01):1450001, 2014.

[25] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1183–1192, Sydney, NSW, Australia, August 2017. JMLR.org.

[26] R. Gallotti and M. Barthelemy. The multilayer temporal network of public transport in Great Britain. *Scientific data*, 2:140056, 2015.

[27] R. Gallotti and M. Barthelemy. The multilayer temporal network of public transport in Great Britain., 2015. https://datadryad.org/resource/doi:10.5061/dryad.pc8m3.

[28] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. W.H. Freeman, New York, 2002.

[29] Steven Gay, François Fages, Thierry Martinez, Sylvain Soliman, and Christine Solnon. On the subgraph epimorphism problem. *Discrete Applied Mathematics*, 162:214–228, 2014.

[30] Yurun Ge and Andrea L Bertozzi. Active learning for the subgraph matching problem. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 2641–2649. IEEE, 2021.

[31] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2018.

[32] W. Han, J. Lee, and J. Lee. Turbo iso: towards ultrafast and robust subgraph isomorphism search in large graph databases. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 337–348. ACM, 2013.

[33] Wook-Shin Han, Jinsoo Lee, and Jeong-Hoon Lee. Turbo iso: towards ultrafast and robust subgraph isomorphism search in large graph databases. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 337–348. ACM, 2013.

[34] H. He and A. Singh. Graphs-at-a-time: query language and access methods for graph databases. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 405–418. ACM, 2008.

[35] X. He. Interactive template map., 2019. https://hexie1995.github.io/transportation-on-the-map/global.

[36] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian Active Learning for Classification and Preference Learning. *arXiv:1112.5745 [cs, stat]*, December 2011. arXiv: 1112.5745.

[37] Vijay Ingalalli, Dino Ienco, and Pascal Poncelet. Sumgra: Querying multigraphs via efficient indexing. In *International Conference on Database and Expert Systems Applications*, pages 387–401. Springer, 2016.

[38] Heinrich Jiang and Maya Gupta. Minimum-Margin Active Learning. *arXiv:1906.00025 [cs, stat]*, May 2019. arXiv: 1906.00025.

[39] Hui Jin, Xie He, Yanghui Wang, Hao Li, and Andrea L Bertozzi. Noisy subgraph isomorphisms on multiplex networks. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4899–4905. IEEE, 2019.

[40] A. Jüttner and P. Madarasi. Vf2++—an improved subgraph isomorphism algorithm. *Discrete Applied Mathematics*, 242:69–81, 2018.

[41] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[42] K. Karra, S. Swarup, and J. Graham. An empirical assessment of the complexity and realism of synthetic social contact networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3959–3967, 10 2018.

[43] A. Kopylov and J. Xu. Filtering strategies for inexact subgraph matching on noisy multiplex networks. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4906–4912, 2019.

[44] L. Kotthoff, C. McCreesh, and C. Solnon. Portfolios of subgraph isomorphism algorithms. In *International Conference on Learning and Intelligent Optimization*, pages 107–122. Springer, 2016.

[45] Zixun Lan, Ye Ma, Limin Yu, Linglong Yuan, and Fei Ma. Aednet: Adaptive edge-deleting network for subgraph matching. *Pattern Recognition*, 133:109033, 2023.

[46] J. Larrosa and G. Valiente. Constraint satisfaction algorithms for graph pattern matching. *Mathematical Structures in Computer Science*, 12(4):403–422, 2002.

[47] Yongjiang Liang and Peixiang Zhao. Similarity search in graph databases: A multi-layered indexing approach. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 783–794. IEEE, 2017.

[48] Xin Liu, Haojie Pan, Mutian He, Yangqiu Song, Xin Jiang, and Lifeng Shang. Neural subgraph isomorphism counting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1959–1969, 2020.

[49] Mauro Maggioni and James M. Murphy. Learning by active nonlinear diffusion. *Foundations of Data Science*, 1(3):271, 2019. Company: Foundations of Data Science Distributor: Foundations of Data Science Institution: Foundations of Data Science Label: Foundations of Data Science Publisher: American Institute of Mathematical Sciences.

[50] M. Magnani and L. Rossi. The ML-Model for Multi-layer Social Networks. In *ASONAM*, pages 5–12. IEEE Computer Society, 2011.

[51] Eric Malmi, Aristides Gionis, and Evimaria Terzi. Active network alignment: a matching-based approach. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1687–1696, 2017.

[52] C. McCreesh and P. Prosser. A parallel, backjumping subgraph isomorphism algorithm using supplemental graphs. In Gilles Pesant, editor, *Int. Conf. on Principles and Practice of Constraint Programming*, pages 295–312. Springer Int. Publishing, 2015.

[53] Ciaran McCreesh, Patrick Prosser, Christine Solnon, and James Trimble. When subgraph isomorphism is really hard, and why this matters for graph databases. *Journal of Artificial Intelligence Research*, 61:723–759, 2018.

[54] Ciaran McCreesh, Patrick Prosser, and James Trimble. The Glasgow subgraph solver: using constraint programming to tackle hard subgraph isomorphism problem variants. In *International Conference on Graph Transformation*, pages 316–324. Springer, 2020.

[55] J. J. McGregor. Relational consistency algorithms and their application in finding subgraph and graph isomorphisms. *Information Sciences*, 19(3):229–250, 1979.

[56] B. D. McKay and A. Piperno. Practical graph isomorphism, II. *J. of Symbolic Computation*, 60:94–112, jan 2014.

[57] J. D. Moorman, Q. Chen, T. K. Tu, Z. M. Boyd, and A. L. Bertozzi. Filtering methods for subgraph matching on multiplex networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3980–3985, Dec 2018.

[58] Jacob D Moorman, Qinyi Chen, Thomas K Tu, Zachary M Boyd, and Andrea L Bertozzi. Filtering methods for subgraph matching on multiplex networks. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 3980–3985. IEEE, 2018.

[59] Jacob D Moorman, Thomas Tu, Qinyi Chen, Xie He, and Andrea Bertozzi. Subgraph matching on multiplex networks. *IEEE Transactions on Network Science and Engineering*, 8(2):1367 – 1384, 2021.

[60] C. Nabti and H. Seba. Compact neighborhood index for subgraph queries in massive graphs. *arXiv preprint arXiv:1703.05547*, 2017.

[61] Thien Nguyen, Dominic Yang, Yurun Ge, Hao Li, and Andrea L Bertozzi. Applications of structural equivalence to subgraph isomorphism on multichannel multigraphs. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4913–4920. IEEE, 2019.

[62] Boyan Onyshkevych. Modeling adversarial activity (maa).

[63] Yeonsu Park, Seongyun Ko, Sourav S Bhowmick, Kyoungmin Kim, Kijae Hong, and Wook-Shin Han. G-care: A framework for performance benchmarking of cardinality estimation techniques for subgraph matching. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1099–1114, 2020.

[64] Heather G Patsolic, Youngser Park, Vince Lyzinski, and Carey E Priebe. Vertex nomination via seeded graph matching. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 13(3):229–244, 2020.

[65] J.-C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proc. of the 12th Nat. Conf. on Artificial Intell, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1.*, pages 362–367, 1994.

[66] X. Ren and J. Wang. Exploiting vertex relationships in speeding up subgraph isomorphism over large graphs. *Proceedings of the VLDB Endowment*, 8(5):617–628, 2015.

[67] M. Puck Rombach, Mason A. Porter, James H. Fowler, and Peter J. Mucha. Core-periphery structure in networks. *SIAM Journal on Applied Mathematics*, 74(1):167–190, 2014.

[68] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (3):353–362, 1983.

[69] C. Schwartz. Modeling Adversarial Activity (MAA), 2018. "(2018, Oct 2)".

[70] Francesc Serratosa and Xavier Cortés. Interactive graph-matching using active query strategies. *Pattern Recognition*, 48(4):1364–1373, 2015.

[71] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2012.

[72] H. Shang, Y. Zhang, X. Lin, and J. X. Yu. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. *Proceedings of the VLDB Endowment*, 1(1):364–375, 2008.

[73] C. Solnon. AllDifferent-based Filtering for Subgraph Isomorphism. *Artificial Intell.*, 174:850–864, August 2010.

[74] Christine Solnon. Experimental evaluation of subgraph isomorphism solvers. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 1–13. Springer, 2019.

[75] Christine Solnon, Guillaume Damiand, Colin De La Higuera, and Jean-Christophe Janodet. On the complexity of submap isomorphism and maximum common submap problems. *Pattern Recognition*, 48(2):302–316, 2015.

[76] D. Sussman, Y. Park, C. E. Priebe, and V. Lyzinski. Matched filters for noisy induced subgraph detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2019.

[77] Simon Tong and Daphne Koller. Support Vector Machine Active Learning with Applications to Text Classification. *Journal of Machine Learning Research*, 2(Nov):45–66, 2001.

[78] Thomas K. Tu, Jacob D. Moorman, Dominic Yang, Qinyi Chen, , and Andrea L. Bertozzi. Inexact attributed subgraph matching. *Proc. IEEE Cong. BIG DATA, Graph Techniques for Adversarial Activity Analytics (GTA3 4.0) workshop*, pages 2575–2582, 2020.

[79] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, January 1976.

[80] J. R Ullmann. Bit-vector algorithms for binary constraint satisfaction and subgraph isomorphism. *Journal of Experimental Algorithmics (JEA)*, 15:1–6, 2010.

[81] United Kingdom's national public transport data repository, 2015. https://data.gov.uk/dataset/d1f9e79f-d9db-44d0-b7b1-41c216fe5df6/national-public-transport-data-repository-nptdr.

[82] Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.

[83] Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Darrin Eide, Kathryn Funk, Rodney Kinney, Ziyang Liu, William Merrill, et al. Cord-19: The covid-19 open research dataset. *ArXiv*, 2020.

[84] Qingyun Wang, Manling Li, Xuan Wang, Nikolaus Parulian, Guangxing Han, Jiawei Ma, Jingxuan Tu, Ying Lin, Haoran Zhang, Weili Liu, et al. Covid-19 literature knowledge graph construction and drug repurposing report generation. *arXiv preprint arXiv:2007.00576*, 2020.

[85] L. Wiskott, Norbert Krüger, N. Kuiger, and C. von der Malsburg. Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):775–779, 1997.

[86] Leonard Wörteler, Moritz Renftle, Theodoros Chondrogiannis, and Michael Grossniklaus. Cardinality estimation using label probability propagation for subgraph matching in property graph databases. In *EDBT*, pages 2–285, 2022.

[87] Dominic Yang, Yurun Ge, Thien Nguyen, Denali Molitor, Jacob D Moorman, and Andrea L Bertozzi. Structural equivalence in subgraph matching. *IEEE Transactions on Network Science and Engineering*, 2023.

[88] S. Zampelli, Y. Deville, and C. Solnon. Solving subgraph isomorphism problems with constraint programming. *Constraints*, 15:327–353, 07 2010.

[89] S. Zhang, S. Li, and J. Yang. Gaddi: distance index based subgraph matching in biological networks. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 192–203. ACM, 2009.

[90] Kangfei Zhao, Jeffrey Xu Yu, Qiyan Li, Hao Zhang, and Yu Rong. Learned sketch for subgraph counting: a holistic approach. *The VLDB Journal*, pages 1–26, 2023.

[91] P. Zhao and J. Han. On graph query optimization in large networks. *Proceedings of the VLDB Endowment*, 3(1-2):340–351, 2010.

[92] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining Active Learning and Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.

[93] Jeremy Zucker, Kaushal Paneri, Sara Mohammad-Taheri, Somya Bhargava, Pallavi Kolambkar, Craig Bakker, Jeremy Teuton, Charles Tapley Hoyt, Kristie Oxford, Robert Ness, and Olga Vitek. Leveraging structured biological knowledge for counterfactual inference: A case study of viral pathogenesis. *IEEE Transactions on Big Data*, 7(1):25–37, 2021.