Applications of Structural Equivalence to Subgraph Isomorphism on Multichannel Multigraphs

Thien Nguyen
University of California, Irvine
thienhn4@uci.edu

Dominic Yang, Yurun Ge, Hao Li, Andrea L. Bertozzi
University of California, Los Angeles
domyang, yurun, lihao0809, bertozzi @math.ucla.edu

***Abstract*—Structural Equivalence refers to the ability to exchange two vertices in a graph without changing the structure of the graph. We provide basic definitions and properties applicable to the subgraph isomorphism problem. We show examples of structural equivalence that reduce the size of the search tree for subgraph isomorphism counting and enumeration, applied to multichannel networks.**

***Index Terms*—graph matching, multichannel multigraphs, multiplex network, structural equivalence, graph compression**

I. Introduction

This paper considers subgraph isomorphisms (SI): given a world graph and a smaller template graph, the goal is to find a copy of the template graph within the world graph. This problem, due to its simple description, finds application in a wide variety of areas. The most immediate applications occur in the fields of biology and chemistry where certain networks of cells, proteins, or molecules are searched for the presence of specific substructures [6] [7] [11]. Subgraph isomorphism has also been used in the field of pattern recognition [3] for problems such as handwriting recognition, facial recognition, and 2D and 3D image analysis. Another application is social network analysis [9].

The subgraph isomorphism problem is NP-complete [2], suggesting that there is no obvious efficient and broadly applicable algorithm, and that solving subgraph isomorphism on graphs is in general computationally intractable. In spite of this difficulty, significant progress has been made in the development of algorithms for detecting subgraph isomorphism. As in other NP-complete problems, the literature addressing the enumeration of exact subgraph isomorphisms has focused on performing a full tree search of the solution space. The state-of-the-art algorithms optimize the variable ordering and pruning of branches of the search tree.

The first significant subgraph isomorphism algorithm proposed by Ullmann [1] generates candidate vertices from the neighbors of already matched world vertices. The widely used VF2 algorithm [4] improves on this by choosing a match ordering that favors template vertices adjacent to already matched template vertices and adding pruning rules based on the degrees of vertices. The authors more recently published the VF3 algorithm [15] that further extends this approach.

In a different approach, Solnon [8] draws upon constraint propagation from artificial intelligence, and her algorithm internally stores candidate lists for every template vertex. By way of repeated application of filters interspersed into a tree search, her algorithm can effectively prune branches to enumerate solutions. She expands on this work in [14] to incorporate more powerful filters.

Significant work has been done to exploit symmetry to compress graphs and count isomorphisms. TurboIso [10] exploits basic symmetry in the template graph and optimizes the matching order based on a selection of candidate regions and exploration within those regions. CFL-Match [13] proposes a match ordering based on a decomposition of the template graph into the core, a highly connected subgraph, and a forest which is further decomposed into a forest and leaves. It also exploits the symmetry in the template graph. BoostIso [12] exploits symmetry in the world graph and presents a method by which other tree-search-based approaches may be accelerated by using their methodology.

Our work is part of the DARPA Modeling Adversial Activity (MAA) program [21]. The data referenced in this paper are synthetically generated by three teams of researchers (Pacific Northwest National Laboratory [17], the GORDIAN (Graphing Observables from Realistic Distributions in Activity Networks) project [18], and Ivysys Technologies [16]). In practice, the examples developed by these teams possess symmetry that results in computational complexity when applying standard subgraph isomorphism approaches. For example, consider the graph in Fig. 1. Vertices of the same color may be interchanged without changing the structure of the graph, implying that any isomorphism can generate many more, simply by swapping vertices. It is our goal to exploit this knowledge to fully determine all possible solutions.

This paper builds upon previous work in [19] and develops a theory of equivalence for both the template and world graph in order to accelerate isomorphism identification. We wish to adequately characterize the larger solution space and present a compact solution representation that can be orders of magnitude smaller than the total count.

This material is based on research sponsored by the Air Force Research Laboratory and DARPA under agreement number FA8750-18-2-0066. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory and DARPA or the U.S. Government. 978-1-7281-0858-2/19/31.00 © 2019*IEEE*

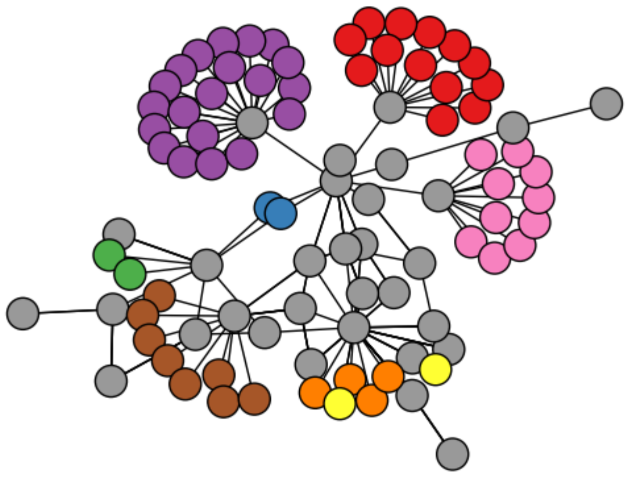

Fig. 1. Example template graph from Ivysys Technologies [16]. Non-gray vertices of the same color may be interchanged without changing the structure of the graph.

II. Definitions and Terminologies

A *graph* G is an ordered pair $\big(V(G), E(G)\big)$ where $V(G)$ is the set of *vertices* or *nodes* of a graph, and $E(G) \subseteq V(G) \times V(G)$ is the set of *edges*. Graphs of this form are referred to as *simple directed graphs*. We only consider directed graphs in this paper, so if $(u, v) \in E(G)$, this does not necessarily imply $(v, u) \in E(G)$. We also consider additional structure in the form of multiedges (or parallel edges) in separate channels. In order to describe these additional features, given a fixed ordering of the vertices $v_1, \dots, v_{|V(G)|}$, we introduce the adjacency matrix for a graph A which is a $|V(G)|$ by $|V(G)|$ matrix where the entry $A[i, j] \in \mathbb{N}$ denotes the number of edges that start at vertex v_i and end at vertex v_j. For graphs with multiple channels, we have multiple adjacency matrices $A^1, \dots, A^m$, one for each channel describing the adjacency structure in that channel, with the non-superscripted matrix A referring to the tuple $(A^1, \dots, A^m)$ of all channels. In general, we use superscripts to denote a graph property for a specific channel. The notation A^c_G denotes the adjacency matrix of a graph G in channel c. We may omit G if the graph is clear from the context. In this paper, we may also slightly abuse notation by writing the number of edges going from vertex u to vertex v in channel c as $A^c[u, v]$, and the tuple of number of edges across all channels $A[u, v] = (A^1[u, v], \dots, A^m[u, v])$. Given this, we let $(u, v) \in E(G)$ if $A[u, v] \neq (0, \dots, 0)$.

We say $H = \big(V(H), E(H)\big)$ is a *subgraph* of G if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. An *independent set* of vertices is a subset of $V(G)$ where there are no edges between any two nodes in the subset. In other words, it is a completely disconnected subgraph. A *clique* in channel c is a set C of nodes such that for all $u \neq v \in C$, $A^c[u, v] > 0$. We say C forms an *n-connected clique* if $A^c[u, v] = n$ for all $u \neq v \in C$ and some positive integer n. That is each pair of nodes in C is connected to each other by exactly n edges. This becomes important later as a property of structural equivalence. Finally, we denote the set of *neighbors* of a vertex v by $N(v) = \big\{u \in V(G) \mid (v, u) \text{ or } (u, v) \in E(G)\big\}$. Similarly, we define the set of incoming neighbors, $N_{in}(u)$ and outgoing neighbors, $N_{out}(u)$.

The central topic of interest in this paper is the problem of finding all subgraph isomorphisms.

Definition 1 (*Subgraph isomorphism*)**.** *Given graphs $T = (V(T), E(T))$ and $W = (V(W), E(W))$, and a map $f : V(T) \longrightarrow V(W)$, we say that f is a subgraph isomorphism if it is injective and edge-preserving, that is, for each $(u, v) \in E(T)$, we have that $(f(u), f(v)) \in E(W)$. In the case of multichannel multigraphs, we need $A^c_T[u, v] \leq A^c_W[f(u), f(v)]$ in each channel c.*

Related terms to this are *subgraph homomorphism* which relaxes the injectivity requirement, and the *induced subgraph isomorphism* which also requires it to be non-edge preserving (if $(u, v) \notin E(T), (f(u), f(v)) \notin E(W)$, and equality in the multichannel multigraph case).

We denote the graph T above as the *template graph* and the graph W as the *world graph*. Any subgraph of W which is isomorphic to T is labeled as a *signal* of T in W. Given a pair of graphs T, W, we say that the pair is *satisfiable* under the subgraph isomorphism problem if there exists at least one signal of T in W. In order to identify potential signal nodes for template vertices, we introduce candidate sets. A candidate set is the set of world vertices that could correspond to a template vertex under an isomorphism map. The candidate set for each template node evolves during steps of the algorithm. Formally, we define a *candidate set* for a vertex $v \in V(T)$ to be any subset of $V(W)$. We denote the candidate set of v as $C(v)$. We say that $C(v)$ is a *minimum complete candidate set* if it contains only $w \in V(W)$ for which there exists some subgraph isomorphism f such that $f(v) = w$. Having introduced appropriate terminologies and notations, we turn to the main topic of this paper.

III. Structural Equivalence

Structural Equivalence is well-known for modeling social networks [5]. Below we will give a motivating example and show several facts that can be useful in computing subgraph isomorphisms. Intuitively, we say that two vertices are structurally equivalent to each other if they can be "swapped" without changing the graph structure. Consider the example in Fig. 2. We see that vertices B and C in the template graph and 2 and 3 in the world graph are interchangeable.

With regards to subgraph isomorphisms, we see that mapping A to 1, B to 2, and C to 3 gives a subgraph isomorphism from the template to the world graph and then swapping the mapping between B and C will give another subgraph isomorphism. Similarly, the same phenomenon happens when we map B and C to 3 and 4: permuting the mapping of the equivalent vertices gives us another subgraph isomorphism. We wish to investigate this property and other usages in the subgraph isomorphism problem.

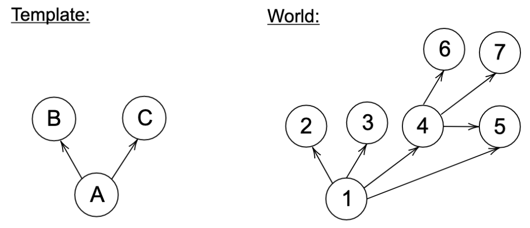

Fig. 2. Running Example

A. Definitions

Structural equivalence occurs when two vertices have the same exact connections to the same set of neighbors. We give a formal definition and show that it gives an equivalence relation on vertices.

Definition 2 (*Structural Equivalence*). *Let G be a graph with n vertices, and x, y be vertices of G. We say that x is structurally equivalent to y in channel c, denoted $x \sim_c y$, if they have:*

- *Same columns and rows: $A^c[x,u] = A^c[y,u]$ and $A^c[u,x] = A^c[u,y]$ for all $u \in V(G) - \{x,y\}$,*
- *Same intersecting entries: $A^c[x,x] = A^c[y,y]$ and $A^c[x,y] = A^c[y,x]$.*

Proposition 3. *$\sim_c$ is an equivalence relation.*

Proof. Given a graph G with n vertices. To simplify notation, let $A^c_G = A$ and $\sim_c = \sim$. We show reflexivity, symmetry, and transitivity of $\sim$. Let $x, y, z \in V(G)$:

Reflexivity: $x \sim x$ – clearly, as x has the same column, row, and intersecting entry ($A[x,x]$) with itself.

Symmetry: Since everything in the definition is symmetrical, $x \sim y$ implies $y \sim x$.

Transitivity: Suppose $x \sim y$ and $y \sim z$. We show $x \sim z$. First, same intersecting entries:

- $A[x,x] = A[y,y] = A[z,z]$ is clear.
- $A[x,z] = A[y,z] = A[z,y] = A[z,x]$, where the first and third equality is due to same rows and columns from $x \sim y$, and the second equality is due to same intersecting entries from $y \sim z$.

Same column: We have $A[u,x] = A[u,y] = A[u,z]$ for all $u \notin \{x,y,z\}$ since $x \sim y$ and $y \sim z$. It remains to show $A[y,x] = A[y,z]$. We have $A[y,x] = A[z,x] = A[x,z] = A[y,z]$, where the first equality is due to same row of $y \sim z$, the second equality due to same intersecting entries shown above, and the third equality due to $x \sim y$ and the same row property. The proof is similar for same row. We have $x \sim z$ as desired. □

Now we generalize this equivalence relation to the multichannel case:

Definition 4 (*Multichannel Equivalence, $\simeq$*). *We say $x \simeq y$ if $x \sim_c y$ for all channel c.*

"$\simeq$" is an equivalence relation because the intersection of equivalence relations is an equivalence relation. From the equivalence relation, we can generate a partition of the template graph. Given any template vertex $x \in V(T)$, we use $[x]$ to denote its equivalence class.

B. Basic Properties

Proposition 5. *Nodes in the same equivalence class either form a k-connected clique (for some $k \in \mathbb{N}$) or an independent set in each channel.*

Proof. It suffices to show that all the nodes in an equivalence class have the same connections between each other in each channel. Given an equivalence class $[x]$, for all $u, v \in [x]$, $u \sim_c v \implies A^c[u,v] = A^c[v,u]$ which establishes our claim. □

We show an important fact that illustrates the usefulness of structural equivalence: given a subgraph isomorphism, we obtain another subgraph isomorphism by switching the images of two equivalent vertices.

Theorem 6 (*Properties of Equivalent Vertices in Subgraph Isomorphism*). *Given a template graph T and a world graph W with m channels, let $f : V(T) \to V(W)$ be a subgraph isomorphism from T to W. Let $x \simeq y$ be two structurally equivalent vertices in $V(T)$. We define $f' : V(t) \to V(w)$ as:*

$$f'(a) = \begin{cases} f(x) & \text{if } a = y, \\ f(y) & \text{if } a = x, \\ f(a) & \text{otherwise.} \end{cases}$$

Then f' is also a subgraph isomorphism from T to W.

Proof. Since f is injective, it is clear that f' is also injective. To see that f' is edge-preserving, we show for all $a, b \in V(T)$, $A_T[a,b] \le A_W[f'(a), f'(b)]$. Without loss of generality, there are three cases:

(i) $a, b \notin \{x,y\}$,
(ii) $a \in \{x,y\}$ and $b \notin \{x,y\}$,
(iii) $a, b \in \{x,y\}$.

For $a, b \notin \{x,y\}$, $(f'(a), f'(b)) = (f(a), f(b))$, and $A_T[a,b] \le A_W[f(a), f(b)]$ satisfies the condition since f is a edge-preserving.

For the second case, WLOG, suppose $a = x$. Then $(f'(a), f'(b)) = (f'(x), f'(b)) = (f(y), f(b))$, and $A_T[a,b] = A_T[x,b] = A_T[y,b]$ due to $x \simeq y$. But we have $A_T[a,b] = A_T[y,b] \le A_W[f(y), f(b)] = A_W[f'(a), f'(b)]$, since f is an isomorphism.

For the final case, WLOG, suppose $a = x$ and $b = y$. We have $A_T[a,b] = A_T[x,y] = A_T[y,x] \le A_W[f(y), f(x)] = A_W[f'(x), f'(y)] = A_W[f'(a), f'(b)]$, where the second equality is due to $x \simeq y$.

These cases show that f' is edge-preserving and moreover, injective. Therefore, it is a subgraph isomorphism. □

This theorem has several major consequences that can significantly reduce redundant computations in highly symmetrical template graphs in that instead of having to consider all candidates permutations of equivalent template nodes, we only have to consider subsets of them, since any matching (of subsets of candidates) gives us that all the permutations are also subgraph isomorphisms. This fact gives a lower bound on the number of subgraph isomorphisms in satisfiable problems.

Corollary 7 (*A lower bound for the number of subgraph isomorphisms*)**.** *Given a satisfiable subgraph isomorphism problem with template T and world graph W, let $P = \{V_1, \ldots, V_n\}$ be a partition of $V(T)$ into structural equivalence classes. Then:*

$$\prod_{i=1}^{n}(|V_i|!)$$

gives a lower-bound for the number of subgraph isomorphisms of T into W.

We proceed by giving a basic modification of the typical backtracking algorithm to apply the previous result.

C. Application to Tree Search

To demonstrate how knowledge of the equivalence structure on a graph can be used for subgraph isomorphisms, we elaborate on tree searches. A tree search in this context involves the construction of a partial match of template vertices to world vertices, at each step extending the matching by assigning the next template vertex to one of its candidate world vertices. If at any point, the matching cannot be extended (due to a contradiction or finding a complete matching), the last made assignment is undone and reassigned to the next candidate vertex. This procedure can be viewed as exploring a tree where each possible assignment of template vertex to world vertex corresponds to a node in the tree, and a path from the root of the tree to a leaf corresponds to a full mapping of vertices.

We display the general purpose procedure as a recursive routine in Algorithm 1. In this procedure, we maintain a binary $|V(T)|$ by $|V(W)|$ matrix C where $C[i,j]$ is 1 if world vertex j is a candidate for template vertex i and 0 otherwise. The function ApplyFilters exists to eliminate candidates and ensure that the next candidate vertex chosen can properly extend the current partial match.

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:         ▷ Add this for loop if using template equivalence
13:         for unmatched $u' \sim u$ do
14:             Set cands$[u', v] = 0$
15:     Let cands = cands_copy
16:     return
```

To demonstrate how template equivalence can accelerate the tree search, we note that by Theorem 6, we can swap the assignments of equivalent template vertices to find another isomorphism. We assert that if we have a partial match, template vertices $u_1 \sim u_2$, and we have just finished trying candidate w for u_1, we do not need to explore branches where u_2 is mapped to w since we can generate any possible isomorphism by taking one where u_1 is mapped to w and swapping. Lines 13 and 14 demonstrate how we can incorporate this fact into a tree search (without these, we would have a standard tree search).

Note that after performing the tree search, the solutions found will be representatives for multiple solutions. Some bookkeeping will need to be done to determine what assignments can be swapped to count the number of distinct solutions. We now generalize the notion of subgraph isomorphism to equivalence classes and apply it to world nodes.

IV. Equivalence Subgraph Isomorphism and Candidate Equivalence

The world graph can also be compressed to avoid redundant branching during tree search. We have already laid out a basic framework for backtracking using equivalence template vertices. Computing equivalence classes for world nodes is trickier because the world graph might contain equivalence relations that are hidden by extra edges. So we do not blindly compute structural equivalence partition on the world graph as that can be expensive and not so useful. Instead, we consider the relationship between candidates at each level of the search-tree. To do so, we compress the template graph using structural equivalence and adapt subgraph matching using an auxiliary data structure called the *candidate structure* instead of the original template graphs.

A. Compressing Graphs with Structural Equivalence

Given a graph G, we construct a new graph $\tilde{G}$ with the same number of channels, called the *equivalence graph* of G, where the vertices of $\tilde{G}$ are the equivalence classes of $V(G)$, called *super-nodes*. We say that a super-node is *non-trivial* if it is an equivalence class of size greater than one. We call the edges between super-nodes *super-edges*, and we use similar notation $V(\tilde{G})$ and $E(\tilde{G})$ to refer to the set of super-nodes and super-edges. The adjacency matrix of $\tilde{G}$ in channel c, $A^c_{\tilde{G}}$ is constructed by setting $A^c_{\tilde{G}}[[u],[v]] = A^c_G[u,v]$. We must show that the definition for super-edges is well-defined.

Proposition 8 (*Characteristics of super-edges*)**.** *Given two super-nodes $[u]$ and $[v]$ in $V(\tilde{G})$, for all vertices $a \in [u]$ and $b \in [v]$, $A_G[u,v] = A_G[a,b]$. Consequently, $A_{\tilde{G}}\left[[u],[v]\right]$ is well-defined.*

Proof. We have $A[u,v] = A[a,v] = A[a,b]$, where the first equality is due to $u \simeq a$, and the second equality is due to $b \simeq v$. □

Given a set of super-nodes S that is a subset of $V(\tilde{G})$, we define the *induced subgraph of $\tilde{G}$*, $\tilde{G}[S]$, similarly to how we would for an uncompressed graph: keeping the super-nodes of

$\tilde{G}$ that contains super-nodes in S and the super-edges between them. An example of the compressed template graph of the running example is shown on the left side of Fig. 3.

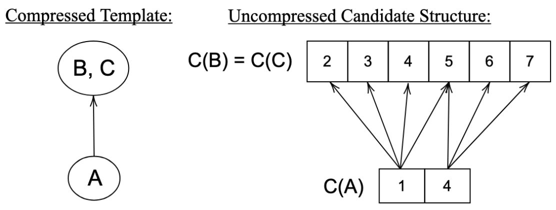

Fig. 3. Compressed Template Graph with Uncompressed Candidate Structure

B. Candidate Structure

We generalize the candidate structure introduced in [20] to the compressed graph of a multichannel multigraph. An uncompressed version is shown on the right of Fig. 3.

Given a super-node $[u]$, a *candidate node* of $[u]$ is a subset of size $|[u]|$ of $C(u)$. The set of all candidate nodes of $[u]$ is denoted as $C([u])$, and we use the notation $v_{[u]} \in C([u])$ to refer to a specific candidate node of $[u]$. Note that $C([u])$ is the set of $|[u]|$-subsets of $C(u)$. This definition is well-defined since it can easily be shown that equivalent template vertices have the same candidates. We use $V(\tilde{W})$ to denote all the candidate nodes of a given world graph and an equivalent template graph. This is a generalization of candidate nodes in the single vertex case to equivalent classes. We refer to the vertices that candidate nodes contain as *world nodes* of a certain candidate node. For example, if $|[u]| = n$ and $v_{[u]} = \{v_1, \dots, v_n\} \in C([u])$, then the v_i's are the world nodes of $v_{[u]}$.

Definition 9 (*Candidate Edges*)**.** *The set of candidate edges in channel i, denoted $CE_i(\tilde{W})$, consisting of pairs of candidate nodes is constructed as follows: For any pair of candidate nodes $a_{[u]} \in C([u])$ and $b_{[v]} \in C([v])$ where $[u], [v] \in V(\tilde{T})$, $(a_{[u]}, b_{[v]})$ is in $CE_i(\tilde{W})$ if $([u],[v]) \in E_i(\tilde{T})$, $a_{[u]} \cap b_{[v]} = \varnothing$, and $a' \in a_{[u]}$ and every world node $b' \in b_{[v]}$, $A_W[a', b'] \geq A_{\tilde{T}}[[u],[v]] = A[u,v]$.*

These conditions enforce that world nodes in the candidate nodes have at least the same connections as the super-edges from the super-nodes in the template graph, and that an edge cannot connect two candidate nodes that have some world nodes in common. We do this because our goal is to only keep the relevant information for the subgraph isomorphism problem.

A *candidate structure* of a template graph T on a world graph W, denoted $\tilde{W}_{\tilde{T}}$ or simply $\tilde{W}$, is a tuple that consists of the candidate nodes $V(\tilde{W})$ and candidate edges $CE(\tilde{W})$ corresponding to super-nodes of $\tilde{T}$. Note that with the given definition of candidate edge, the candidate structure is a simple directed graph. For an example, consider the candidate structure of the running example is presented in Fig. 4 after obtaining the candidates of $A, B,$ and C from applying the statistics and topology filter from [19]. The candidate nodes

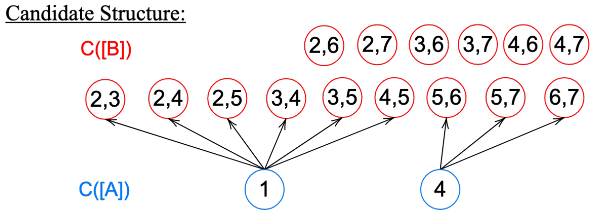

Fig. 4. Candidate Structure of the Running Example

of $C([A])$ are shown in blue and $C([B]) = C([C])$ in red. We note that some of the candidate nodes like $\{2,6\}$ through $\{4,7\}$ do not have a candidate edge because they do not meet the requirement in definition 9. From the candidate structure graph for this example, we can almost read off the isomorphisms and remove candidates who do not have appropriate candidate edges corresponding to template edges (a variant of the topology filter in [19]).

C. TE Subgraph Isomorphism

Having developed the terminology and tools for incorporating equivalence classes into the subgraph isomorphism (SI) problem, we will now give a formal definition of subgraph isomorphism using equivalent template graphs.

Definition 10 (*TE Subgraph Isomorphism*)**.** *Given a template graph T and a world graph W, let $\tilde{T}$ be the equivalence graph of T. We say a map $f : V(\tilde{T}) \longrightarrow \mathcal{P}(W)$ is an TE (template equivalence) subgraph isomorphism from $\tilde{T}$ to W if f satisfies the following conditions:*

(i) for all $[u],[v] \in V(\tilde{T})$, $[u] \neq [v]$, $f([u]) \cap f([v]) = \varnothing$, which implies that f is injective,
(ii) for all $[u] \in V(\tilde{T})$, $|[u]| = |f([u])|$,
(iii) for all $[u] \in V(\tilde{T})$, if $[u]$ forms a k-connected clique in channel i of T, then $f([u])$ must also contain a k-connected clique in channel i of W,
(iv) For all $[u],[v] \in \tilde{T}$, if $([u],[v]) \in E_i(\tilde{T})$ then $(f([u]), f([v])) \in CE_i(\tilde{W})$.

When given a TE subgraph isomorphism, we can obtain a standard subgraph isomorphism by fixing an ordering on each of the template vertices contained within a super-node. The set of all possible subgraph isomorphisms is then generated by considering all possible orderings on each super-node.

We modify the current tree search introduced in Section III-C by adding a new extension rule. We introduce a *joinable* condition for TE subgraph isomorphism.

Definition 11 (*Partial Matches for Equivalence Classes*)**.** *A partial match M of the equivalent graph, $\tilde{T}$, of a template graph T to a world graph W is a map from a subset M_T of $V(\tilde{T})$ to a subset, M_W, of $\mathcal{P}(V(W))$ such that M is a TE SI from $\tilde{T}[M_T]$ to W.*

Remark 12. *A partial match of equivalent classes of size $|V(\tilde{T})|$ is a TE SI from $\tilde{T}$ to W.*

Now we give the conditions for extending a current partial match to a new match and that it forms a new partial match.

Definition 13 (*M-Joinable*). *Given a partial match of equivalence classes M, candidate structure $\tilde{W}$, and a new pair of super-node $[u]$ and candidate node $v_{[u]} \in C([u])$ to match, we say that $([u], v_{[u]})$ is M-Joinable if:*

(i) $[u] \notin M_T$, *and* $\forall([x], a_{[x]}) \in M$, $v_{[u]} \cap a_{[x]} = \varnothing$

(ii) $|[u]| = |v_{[u]}|$,

(iii) For every super-node $[u]$ that forms a k-connected clique in channel i, $v_{[u]}$ must contain a $k-connected$ clique in channel i,

(iv) For every $[u'] \in N([u]) \cap M_T$, we must have:

- $([u], [u']) \in E(\tilde{T}) \implies (v_{[u]}, M([u'])) \in CE(\tilde{W})$,
- $([u'], [u]) \in E(\tilde{T}) \implies (M([u']), v_{[u]}) \in CE(\tilde{W})$.

To use it on our modified tree search, we must show it extends some partial match.

Proposition 14 (*Extending Partial Matches*). *If M is a partial match of equivalence classes and $([u], v_{[u]})$ is an M-Joinable pair, then $M' = M \cup ([u], v_{[u]})$ is another partial match of size $|M_T| + 1$.*

Proof. To show M' is a partial match, we must show that M' is a TE SI from $\tilde{T}[M'_T]$ to W. We check:

(i) Since M is a partial match and $[u] \notin M_T, v_{[u]} \cap M_W = \varnothing$ by 13(i), M' is well-defined and injective. Also, for the same reason, $M'([a]) \cap M'([b]) = \varnothing$ for any $[a], [b]$ in M'_T.

(ii) For all $[a] \in M'_T$, $|[a]| = |M'([a])|$ since $M'([a]) = M([a])$ for $a \notin [u]$ and M is a partial match. The verification for $[u]$ follows from 13(ii).

(iii) The clique condition is specified in 13(iii) for $[u]$ and follows from M being a partial match for all other super-nodes in M_T.

(iv) For all $([a], [b]) \in E_i(\tilde{T}[M'_T])$, if $[a], [b] \neq [u]$, the edge condition is satisfied due to M being a partial match.

WLOG, suppose $[a] = [u]$ i.e. $([u], [b]) \in E_i(\tilde{T}[M'_T])$. We have $(M'([u]), M'([b])) \in CE_i(\tilde{W})$ from condition (iv) in def 13.

This shows that M' is a partial match of $\tilde{T}[M'_T]$ to W. □

In the running example, supposed we have a partial match $[A]$ with 1 from Fig. 4, we can extend this by matching $[B]$ with $\{2, 3\}$ by checking the joinable condition. Since it's joinable, and our partial match is now of size $|V(\tilde{T})|$, we have a subgraph isomorphism. We can use this to easily develop a backtracking-algorithm for equivalence template graphs.

D. Candidate Node Equivalence

We start with an observation that, due to how we traverse the search tree using candidate edges, at every level of the search tree, the subtree explored rooted at a candidate node will be similar to another candidate node if they have the same set of neighboring candidate nodes. We formalize these observations and give several useful applications.

Definition 15 (*Candidate Node Equivalence*). *Given a super-node $[u] \in V(\tilde{T})$, we say that two candidate nodes $a_{[u]}, b_{[u]} \in C([u])$ are equivalent, denoted $a_{[u]} \simeq b_{[u]}$, if they have the same set of neighbors in $\tilde{W}_T$ i.e. $N_{in}(a_{[u]}) = N_{in}(b_{[u]})$ and $N_{out}(a_{[u]}) = N_{out}(b_{[u]})$.*

This relation is clearly another equivalence relation since it is the same equivalence relation as in Definition 2 but on the bigger candidate graph $\tilde{W}$ (where there's no complication of multiedges or self loops). An example can be seen in Fig. 4 where all the nodes adjacent to node 1 are equivalent candidates of $C([B])$, and the set of nodes adjacent to 4 forms another candidate equivalence class. This definition also applies to the uncompressed candidate structure where we just consider candidates of ordinary vertices. An example of this case can be seen in Fig. 3, where nodes $2, 3, 4 \in C(B)$ form a candidate equivalence class, and nodes $6, 7 \in C(B)$ form another. We now state a property that relates candidate equivalence in the uncompressed case to the compressed case which can be useful for computing candidate equivalence for non-trivial super-nodes. Let $[S]^k$ denotes the set of k-subsets of a set S.

Proposition 16. *Given a super-node $[u] \in V(\tilde{T})$, consider $C(u)$ and its candidate equivalence partition $U_1, \ldots, U_m$ in the uncompressed candidate structure. Then any two candidates of $[u]$ constructed by choosing $|[u]|$ nodes in any U_i are equivalent i.e. $a_{[u]}, b_{[u]} \in [\mathcal{P}(U_i)]^{|[u]|} \subseteq C([u])$ implies $a_{[u]} \simeq b_{[u]}$.*

Proof. This can be verified by checking the candidate edge condition and how superedges are formed. □

Note that if $|U_i| < |[u]|$ then $[\mathcal{P}(U_i)]^{|[u]|} = \varnothing$, and hence the proposition only applies to equivalence classes of size greater than $|[u]|$. Now we show the main result that allows us to interchange world vertices given a TE SI.

Theorem 17 (*Interchangeability of equivalent candidate nodes*). *Given a TE subgraph isomorphism $f : V(\tilde{T}) \longrightarrow V(\tilde{W})$ and equivalent candidate nodes $x_{[u]}, y_{[u]} \in C([u])$, if $f([u]) = x_{[u]}$ and $f([v]) \cap y_{[u]} = \varnothing$ for all $[v] \in V(\tilde{T}) - [u]$, then defining f' as:*

$$f'([v]) = \begin{cases} y_{[u]}, & \textit{if } [v] = [u] \\ f([v]), & \textit{otherwise.} \end{cases}$$

gives another TE subgraph isomorphism.

Proof. We note that $f - ([u], x_{[u]}) = f' - ([u], y_{[u]})$ is a partial match (from def 11). It suffices to show $([u], y_{[u]})$ is joinable (by remark 12). Since $f([v]) \cap y = \varnothing$ for all $[v] \in V(\tilde{T})$ and $x \simeq y$, condition (i) to (iii) in definition 13 is satisfied. Also because $x_{[u]} \simeq y_{[u]}$ and how $y_{[u]}$ has all $x_{[u]}$'s neighbors in the given match by definition, condition (iv) is also satisfied. Hence, f' is a TE subgraph isomorphism. □

This property gives us a powerful way to permute now the world nodes in a match. In order to exploit it, we must not only take into consideration the computation the equivalence candidates but also how to deal with the intersecting condition requirement.

E. Non-Intersecting Candidates and a New Algorithm for Equivalent Candidate Nodes

We want to use theorem 17 to avoid redundant branching during a tree search. One simple case is when, during a level of the tree search, the intersection between a candidate equivalence class of a certain template vertex and the candidates of unmatched vertices is *empty*. This would guarantee the validity of the intersecting condition in theorem 17, which means we would only need to consider a representative candidate node in each equivalence class. This gives a guideline to modify the current backtracking algorithm:

1) At every recursive call matching template vertex $[v]$ with candidates, compute the candidate equivalence partition and put the equivalence classes into a queue Q.
2) Take a candidate equivalence class from Q, and check if there's any intersection with the candidates of unmatched template vertices.
3) If the intersection is empty, go to step 4. Otherwise, remove each of the intersecting node from the candidate equivalence class and add them to Q as a single equivalence class. Then go to step 4.
4) Match the candidate equivalence class, $m_{[v]}$, with $[v]$ and repeat while only checking for joinability on a representative candidate.
5) If Q is not empty, go to to step 2. Otherwise, return.

This procedure can significantly reduce the number of branches given a high number of equivalent candidates. Combining theorems 6 and 17, we obtain $\binom{|m_{[v]}|}{|[v]|}|[v]|!$ isomorphisms from each match.

V. Experiments

In order to quantify the speed up and compression factor granted by the use of our equivalence structures, we performed time tests on various data sets provided by DARPA-MAA teams. These data sets are synthetic and have been generated for the purposes of modeling a variety of transactions across various channels, representing phone calls, emails, bank transactions, etc. between human actors. General information about the data sets is included in Table I.

TABLE I
Data Set Information

Data Set	$\lvert V(T)\rvert$	$\lvert E(T)\rvert$	$\lvert V(W)\rvert$	$\lvert E(W)\rvert$	Channels
pnnl_v4	75	857	22154	480242	6
pnnl_v5	73	1554	23037	12393659	7
pnnl_v6	74	1620	22996	12318861	7
pnnl_rw	35	158	6407	74862	3
gordian_v6	161	1850	101934	84854616	8
gordian_v7	156	3045	190869	123267100	11
ivysys_v6	92	195	2488	5470454	3
ivysys_v7	92	195	2488	5470970	3

In each sample template-world pair, we considered three different scenarios: no equivalence (NE), solely template equivalence (TE), and both template and candidate equivalence or full equivalence (FE). In each case, we determined the time it took to enumerate all the solutions (or timed it out at ten minutes if enumerating all solutions took longer than this). We computed the number of solutions found as well as the number of "representative solutions" found corresponding to the actual number of leaves reached in the solution tree. The ratio of these numbers can give a scale of the compression possible by considering equivalence classes and is recorded in the last column of table II. The results for each data set are displayed in Table II. The experiments were performed on an Intel Xeon Gold 6136 processor with 3 GHz, 25 MB of cache, and 125 GB of memory.

TABLE II
Data Solver Statistics

Data Set	Nontrivial Eq. Class Sizes	Alg.	Run Time (s)	Solution Count	Rep. Solution Count	Compr. Rate
pnnl_rw	2	NE	>600	3.85e+8	384648	1
		TE	>600	6.50e+5	325091	0.5
		FE	>600	5.45e+8	5029	9.23e-6
pnnl_v4	2,6	NE	>600	1.43e+5	143380	1
		TE	158.94	5.71e+7	39680	0.0007
		FE	126.78	5.71e+7	112	1.96e-6
pnnl_v5	2,3,6	NE	>600	34230	34230	1
		TE	0.35	181440	21	0.0001
		FE	0.39	181440	1	5.51e-6
pnnl_v6	2,4,4	NE	427.76	1152	1152	1
		TE	0.34	1152	1	0.0009
		FE	0.40	1152	1	0.0009
gordian_v6	2,2,142	NE	>600	1832	1832	1
		TE	1.46	5.18e+247	48	9.28e-247
		FE	0.37	5.18e+247	1	1.93e-248
gordian_v7	2,2,6	NE	>600	68289	68289	1
		TE	>600	1.98e+8	68850	0.0003
		FE	>600	3.80e+11	873	2.30e-9
ivysys_v6	2,2,2,4,8, 10,13,20	NE	>600	9460	9460	1
		TE	>600	2.07e+46	48710	2.35e-42
		FE	>600	8.87e+59	5719	6.45e-57
ivysys_v7	2,2,2,4,8, 10,13,20	NE	>600	40520	40520	1
		TE	>600	2.134e+46	50148	2.35e-42
		FE	>600	7.76e+61	5645	7.27e-59

A. Discussion

As can be seen in all cases considered, there is clear benefit to explicitly addressing the equivalence apparent in subgraph isomorphism problems. Visually, the amount of compression in the template graph is significant as can be seen in Figure 5 which is the equivalent graph version of Figure 1 thus reducing the number of nodes in the template graph from 92 to 39.

In examples such as pnnl_v5, pnnl_v6, and gordian_v6, our method can reduce a monumental number of solutions to a single representative solution. In the case for the PNNL generated data, this captures the method used to generate the problem which is based on embedding a single solution.

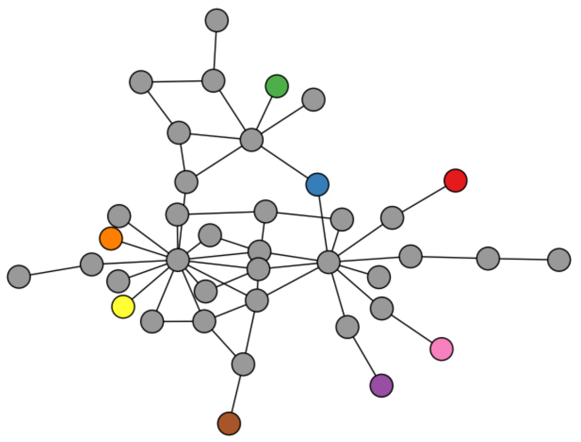

Fig. 5. Compressed version of the graph presented in Figure 1. Each colored node corresponds to the group of nodes of the same color in Figure 1.

In the cases where the solvers timed out, the use of equivalence classes still allowed for the discovery of orders of magnitude more solutions than the NE Solver. For some of the harder data sets like those produced by Ivysys Technologies, we may need even more refined notions of equivalence to capture the full solution space.

VI. Conclusion

In this paper, we exploit the symmetry of the template and world graphs via vertex equivalence, to better understand the full solution space of subgraph isomorphism problems for multichannel networks. These equivalence relations enable us to compress the template and world graphs; this compression significantly reduces the computation time for finding all subgraph isomorphisms and allows for a compact representation of the solution space. We developed efficient computation routines for computing the equivalence relations and adapted the previously developed framework for equivalence classes. We tested this approach on several synthetic DARPA MAA datasets and observed considerable improvement on datasets with several equivalence classes, such as those produced by Ivysys Technologies and GORDIAN.

For future work, one obvious direction is the case where the intersections between candidate equivalence classes and unmatched vertices' candidates are non-trivial. Other directions include inexact subgraph matching, integrating node cover methods, vertex ordering, and further development of filters.

References

[1] J. Ullmann, "An algorithm for subgraph isomorphism," *Journal of the ACM (JACM)*, vol. 23, no. 1, pp. 31–42, 1976.

[2] M. R. Garey and D. S. Johnson, *Computers and intractability*. New York: W.H. Freeman, 2002, vol. 29.

[3] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *International journal of pattern recognition and artificial intelligence*, vol. 18, no. 03, pp. 265–298, 2004.

[4] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub) graph isomorphism algorithm for matching large graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 10, pp. 1367–1372, 2004.

[5] R. A. Hanneman and M. Riddle, *Introduction to social network methods*, 2005.

[6] T. Aittokallio and B. Schwikowski, "Graph-based methods for analysing networks in cell biology," *Briefings in Bioinformatics*, vol. 7, no. 3, pp. 243–255, Sep. 2006.

[7] V. Lacroix, C. G. Fernandes, and M.-F. Sagot, "Motif search in graphs: Application to metabolic networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 3, no. 4, pp. 360–368, 2006.

[8] C. Solnon, "Alldifferent-based filtering for subgraph isomorphism," *Artificial Intelligence*, vol. 174, no. 12-13, pp. 850–864, 2010.

[9] W. Fan, "Graph pattern matching revised for social network analysis," in *Proceedings of the 15th International Conference on Database Theory*, ACM, 2012, pp. 8–21.

[10] W.-S. Han, J. Lee, and J.-h. Lee, "Turboiso: Towards ultrafast and robust subgraph isomorphism search in large graph databases," in *SIGMOD Conference*, 2013.

[11] A. Kuccukural, A. Szilagyi, O. U. Sezerman, and Y. Zhang, "Protein homology analysis for function prediction with parallel sub-graph isomorphism," in *Bioinformatics: concepts, methodologies, tools, and applications*, IGI Global, 2013, pp. 386–399.

[12] X. Ren and J. Wang, "Exploiting vertex relationships in speeding up subgraph isomorphism over large graphs," *Proceedings of the VLDB Endowment*, vol. 8, no. 5, pp. 617–628, 2015.

[13] 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*, ACM, 2016, pp. 1199–1214.

[14] L. Kotthoff, C. McCreesh, and C. Solnon, "Portfolios of subgraph isomorphism algorithms," in *Learning and Intelligent Optimization*, P. Festa, M. Sellmann, and J. Vanschoren, Eds., Cham: Springer International Publishing, 2016, pp. 107–122.

[15] V. Carletti, P. Foggia, A. Saggese, and M. Vento, "Introducing VF3: A new algorithm for subgraph isomorphism," in *International Workshop on Graph-Based Representations in Pattern Recognition*, Springer, 2017, pp. 128–139.

[16] 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)*, Oct. 2018, pp. 3986–3990.

[17] J. A. Cottam, S. Purohit, P. Mackey, and G. Chin, "Multichannel large network simulation including adversarial activity," in *2018 IEEE International Conference on Big Data (Big Data)*, Oct. 2018, pp. 3947–3950.

[18] 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)*, Oct. 2018, pp. 3959–3967.

[19] 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)*, IEEE, 2018, pp. 3980–3985.

[20] M. Han, H. Kim, G. Gu, K. Park, and W.-S. Han, "Efficient subgraph matching: Harmonizing dynamic programming, adaptive matching order, and failing set together," in *Proceedings of the 2019 International Conference on Management of Data*, ACM, 2019, pp. 1429–1446.

[21] B. Onyshkevych, *Modeling adversarial activity (maa)*. [Online]. Available: https://www.darpa.mil/program/modeling-adversarial-activity.