



AFRL-RI-RS-TR-2022-121

VARIATIONAL METHODS ON GRAPHS FOR IDENTIFICATION ON TRANSACTION NETWORKS

UNIVERSITY OF CALIFORNIA, LOS ANGELES (UCLA)

AUGUST 2022

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC)(<http://www.dtic.mil>).

AFRL-RI-RS-TR-2022-121 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /

NANCY A. SEPULVEDA
Work Unit Manager

/ S /

SCOTT D. PATRICK
Deputy Chief
Intelligence Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE		2. REPORT TYPE		3. DATES COVERED	
AUGUST 2022		FINAL TECHNICAL REPORT		START DATE JANUARY 2018	END DATE JANUARY 2022
4. TITLE AND SUBTITLE Variational Methods on Graphs for Identification on Transaction Networks					
5a. CONTRACT NUMBER FA8750-18-2-0066		5b. GRANT NUMBER N/A		5c. PROGRAM ELEMENT NUMBER 62702F	
5d. PROJECT NUMBER		5e. TASK NUMBER		5f. WORK UNIT NUMBER R2GM	
6. AUTHOR(S) Andrea Bertozzi					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of California, Los Angeles (UCLA) 520 Portola Plaza Los Angeles CA 90095				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RIEA 525 Brooks Road Rome NY 13441-4505			10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI		11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-RI-RS-TR-2022-121
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The project is fundamental unclassified research on graph detection methods applied to synthetic data, which was modeled on transaction data and provided to the project by the government. The project is critical to national security because such data is often the primary information known regarding adversaries interested in carrying out adversarial activities. The mathematical structure of the information to be analyzed will be a directed graph with a large number of nodes and even more edges. Additional information may be present including time stamps and meta-data however such information will not be the primary focus on the research. With the graph structure as the primary information, our goal is to detect subgraphs/activities on graphs of a specified known type but with information that potentially has noise, missing information and mis-alignment of the graph.					
15. SUBJECT TERMS Graph detection methods, adversarial activity, subgraph isomorphism, transactional network graphs					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT		18. NUMBER OF PAGES
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	SAR		48
19a. NAME OF RESPONSIBLE PERSON NANCY A. SEPULVEDA					19b. PHONE NUMBER (Include area code) N/A

Table of Contents

List of Figures	ii
List of Tables	iii
1.0 SUMMARY	1
2.0 INTRODUCTION	2
3.0 METHODS, ASSUMPTIONS, AND PROCEDURES	6
4.0 RESULTS AND DISCUSSION	19
5.0 CONCLUSIONS	33
6.0 REFERENCES	35
7.0 APPENDIX A – Publications and Presentations	37
7.1 IEEE BIG DATA Workshops on Graph Techniques for Adversarial Activity Analytics (GTA3)	39
8.0 LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS	42

LIST OF FIGURES

Figure 1: Program Technical Areas	2
Figure 2: Exact matching algorithm flowchart	13
Figure 3: Flowchart for inexact subgraph matching code	14
Figure 4 AIDA V2.0.2 template graph(left) and subgraph of world graph with nodes participating in minimal alignable subgraph isomorphism(right).....	17
Figure 5: A biochemical reactions template graph (left) and the solution-induced world subgraph (right) for a solution class comprised of 9.18×10^{13} solutions.....	18
Figure 6: Number of candidates for each template node(top) and number of template nodes for which each world node is a candidate(bottom).....	20
Figure 7: a diagram for the average compression rate for the single channel benchmark networks above using different algorithms for each notion of equivalence	21
Figure 8: Diagram of the solution space for Template 1B for the AIDA v2.1.2 dataset.....	25
Figure 9: AIDA 2.1.2 Comparison of our results to the results from other TA3 performers in the DARPA MAA Program.....	26
Figure 10: Template 1 is a structural template describing IL-6 amplification, a proposed mechanism for how COVID-19 induces cytokine storms.	29
Figure 11: Template for COVID paths Dataset version 3.3.1.	31
Figure 12: COVID-19 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.	32

LIST OF TABLES

Table 1. Results of our exact matching code for the Phase I DARPA MAA datasets.	19
Table 2. Benchmark datasets for single channel subgraph match study.....	21
Table 3: multichannel datasets on which we tested structural equivalence algorithms with the results on the right.....	22
Table 4: Run times to find all the subgraph isomorphisms using different notions of equivalence.	22
Table 5: Results from our inexact matching code for AIDA v2.1.2.	23
Table 6: Aida v. 2.1.2 from Phase II of the MAA program.....	24
Table 7: Number of solutions found for each template of AIDA 2.1.2.	27

1.0 SUMMARY

The project is fundamental unclassified research on graph detection methods applied to synthetic data, which was modeled on transaction data and provided to the project by the government. The project is critical to national security because such data is often the primary information known regarding adversaries interested in carrying out adversarial activities. The mathematical structure of the information to be analyzed will be a directed graph with a large number of nodes and even more edges. Additional information may be present including time stamps and meta-data however such information will not be the primary focus on the research. With the graph structure as the primary information, our goal is to detect subgraphs/activities on graphs of a specified known type but with information that potentially has noise, missing information and mis-alignment of the graph.

2.0 INTRODUCTION

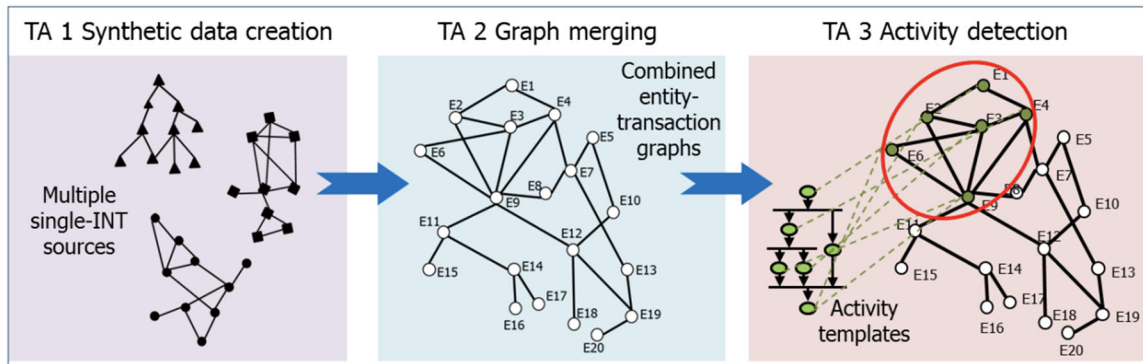


Figure 1: Program Technical Areas

The Defense Advanced Research Projects Agency (DARPA) Program on Modeling Adversarial Activity (MAA) consists of three main areas - the creation of synthetic data for adversarial activities within a large background of information (TA1), the merging and alignment of graph information (TA2), and detection of activities in the merged information space (TA3). This project specifically addressed the TA3 activity detection problem which can be formulated as a subgraph isomorphism problem. Fig 1 above is taken from the DARPA MAA program call.

Subgraph matching is studied in the computer science literature, mostly for the single channel case. Most state of the art algorithm approaches for subgraph isomorphism follow one of three approaches [Carletti et al., 2017]: tree search, constraint propagation, and graph indexing.

Tree search approaches keep track of a search state, and navigate the tree of possible search states, backtracking when they reach the end of a branch. Due to the enormity of this tree, to limit computational complexity as much as possible, these approaches refine the search space at each step of the search to avoid unnecessary branches. Examples of tree search approaches include Ullmann's algorithm [Ullmann, 1976], VF2 [Cordella et al., 2004] and its variants (VF2Plus [Carletti et al., 2015], VF3 [Carletti et al., 2017], VF2++ [Jüttner and Madarasi, 2018]), and for specific graphs, RI/RI-DS [Bonnici et al., 2013].

Constraint propagation approaches view subgraph isomorphism as a constraint satisfaction problem, where variables are assigned values while satisfying a given set of constraints. These approaches keep track of a compatibility matrix that indicates which nodes in the world are possible matches for the template nodes. By repeatedly applying local constraints, this matrix is reduced until only a few possible candidate matches remain. The matrix can then be explored to find all solutions. Examples of constraint propagation approaches include McGregor [McGregor, 1979], nRF+ [Larrosa and Valiente, 2002], ILF [Zampelli et al., 2010], LAD [Solnon, 2010] (and its variants, IncompleteLAD and PathLAD [Kotthoff et al., 2016]), McCreesh and Prosser (Glasgow) [McCreesh and Prosser, 2015], and FocusSearch [Ullmann, 2010].

Graph indexing approaches seek to retrieve from a database of graphs all graphs that match a given subgraph query. To accelerate searching, they construct indexes for the database, so that future

searches will be efficient. These indexes are often based on characteristic substructures of the template. A Cartesian product is then performed on the results of these indexed queries, identifying all possible matches. Next, there is a verification step to check which retrieved graphs fully match the pattern; this typically involves running another subgraph isomorphism algorithm, such as VF2. Examples of graph indexing approaches include GraphQL [He and Singh, 2008], SPath [Zhao and Han, 2010], GADDI [Zhang et al., 2009], QuickSI [Shang et al., 2008], TurboISO [Han et al., 2013], BoostISO [Ren and Wang, 2015], CFL-Match [Bi et al., 2016], and CNI-Match [Nabti and Seba, 2017]. Though our problem does not involve graph databases and we thus do not construct indices, several of these approaches involve filtering techniques which can be used independently of the indices they construct. [Sun and Luo, 2019].

2.1. Ullmann’s Algorithm

Ullmann’s algorithm [Ullmann, 1976] is a backtracking tree search with refinement. For every template node t , they create a list of candidate nodes in the world which could correspond to t . Initially, this is simply all nodes with degree greater than or equal to the degree of t . At each step in the tree search, a candidate node is chosen as a match for t . To reduce computation time, the remaining candidates are then refined as follows. For every pair of template nodes joined by an edge, their candidates should also be joined by an edge in the world. Any candidate for one of these connected nodes that does not have an edge to any candidate for the other connected node can be removed from consideration. Our topology filter is a simple extension of this constraint to the multiplex network case.

2.2. VF2

VF2 [Cordella et al., 2004] generalizes Ullmann’s algorithm to directed graphs and extends refinement with additional semantic feasibility rules. They distinguish matched nodes (nodes with only one candidate) from other nodes and ensure that candidates have more matched neighbors and unmatched neighbors than their corresponding template node; in the directed case, these neighbors must have edges with matching directions. They also enforce a matching order where neighbors of previously matched nodes are matched before other nodes.

2.3. Constraint Propagation Approaches

In [Larrosa and Valiente, 2002], the authors classify existing constraints using two categorizations: binary vs. non-binary and forward checking (FC) vs. really forward look ahead (RF). Binary constraints map N variables to m values using an $N \times M$ matrix of binary-valued variables, while non-binary constraints use a vector of length n , whose entries are restricted to m values. The authors propose $nRF+$, a non-binary really forward lookahead algorithm, with specific look ahead for subgraph isomorphism. They enforce the constraint that if v is a candidate for template node u , then the number of u ’s neighbors must be less than or equal to the number of candidates for those neighbors that are adjacent to v . The paper [Solnon, 2010] classifies many constraints previously used in different subgraph isomorphism algorithms. The author proposes a new algorithm, LAD, using the constraint that for a match to exist between nodes m and n , there must exist a matching between their neighborhoods subject to the *alldifferent* constraint, which can be identified using the Hopcroft-Karp algorithm. This forms the basis for our neighborhood filter.

2.4. Graph Indexing Approaches

In GraphQL [He and Singh, 2008], the authors describe a graph query language and provide an algorithm for resolving graph database queries. In particular, they iterate a constraint similar to that of [Solnon, 2010], where there must exist a bipartite matching between the neighborhoods of a template node and its candidate. This approach is further expanded upon in SPath [Zhao and Han, 2010], where the k-distance neighborhood is also considered. TurboISO proposes a method for handling permutable template nodes, as well as addressing the order of matching when identifying isomorphisms. They construct an NEC tree, whose vertices represent groups of permutable template nodes, and perform matching using this tree. At the end, they combine and permute the candidates for each of these template nodes. For the matching order, they divide the graph into candidate subregions, each of which contains a group of candidates that may take part in a signal. They then prioritize searching the smaller candidate subregions. We take a similar approach, prioritizing nodes with the least number of candidates.

In CFL-Match [Bi et al., 2016], the template is first decomposed into three types of structures: Core, Forest, and Leaf. To do this, they first construct a spanning tree of the template. Then, they compute the minimal connected subgraph containing all nontree edges of the template. They then iteratively remove all nodes with degree 1, updating the degree counts after each removal. The remaining nodes form the core. For each node in the core that is connected to a non-core node, a forest structure is created, consisting of that node and all non-core nodes it is connected to. Finally, the leaf structure consists of chains of removed non-core nodes. After decomposing the template in this way, each structure is queried independently. In order to postpone the Cartesian product of the results as much as possible, the core is queried first, and the results are then used to restrict queries for each forest structure. Similarly, the leaf queries are restricted by the results of the forest queries.

Most existing subgraph isomorphism approaches in the literature do not address the multigraph case. SuMGrA [Ingalalli et al., 2016] details a graph indexing and backtracking tree search approach for the multigraph case, and RI [Bonnici et al., 2013] can be extended to handle multigraphs.

2.5 Tasks for our project

Our project addressed three main tasks:

Task 1. Variational models for activity detection: We construct variational models for activity detection on graphs. We will consider two kinds of graphs - the network graphs (provided by the government) and additional graphs identified by our research team as relevant to the project. The methods are primarily built around and focused on network graphs.

Task 2. Performance analysis and detectability bounds: We planned to generate rigorous theory for performance of our methods and for detectability. Once we realized that the solution space is combinatorially complex, we modified Task 2 to develop rigorous theory to understand the solution space and to further use this knowledge to increase the capture of subgraph isomorphisms.

Task 3. Highly scalable merging and detection algorithms: We focused on methods that are provably scalable, with a target demonstrating the ability to handle state of the art challenge problems posed by the TA1 and TA2 performers.

2.6 Deliverables: Our deliverables included a suite of scalable algorithms for robust activity detection on merged graphs, as well as a set of concrete theoretical results on the properties of these algorithms. We also delivered results of our algorithms on datasets provided by DARPA MAA TA1 performers. We have published numerous research papers in the public domain about this problem.

We have released all components of this project under an appropriate open source license and make all of the code available on GitHub or a similar source. This ensures that the algorithms developed are easily accessible to potential users/developers who will support and maintain the project over time.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

3.1 Task 1

3.1.1 Phase I - exact subgraph matching based on graph structure only

Many subgraph matching algorithms apply a branch-and-bound approach to search the space of all subgraphs of the background. They grow a partial match between the template and a subgraph of the background, adding one or more nodes at a time to the partial match using various heuristics and backtracking if a complete match becomes impossible. What differentiates subgraph matching algorithms of this type is how they decide what order to add nodes to the partial match, and how they decide which background nodes are considered to be *candidates* for each template node given a partial match. The latter process, that of finding candidate background nodes for each template node, is called *filtering*. It is also sometimes called pruning. Some references from the literature are listed in the references section.

Our first step in searching for signals in a large background is to efficiently reduce the number of background nodes which must be considered. To do this, we seek to identify candidates for each of the template nodes and eliminate background nodes that are not candidates for any template node. We refer to this process as filtering. The filter consists of two stages: (1) via node-level statistics; (2) via topology. In the first stage we compare node-level statistics for background nodes against corresponding statistics of template nodes. In the second stage, we utilize the candidate lists from the first stage to enforce local topological structure of the template. We have examples where this approach works well and other examples where the combinatorial complexity of the solution space requires looking at an additional permutation filter.

We empirically show that a combination of three efficient filtering methods is highly effective at reducing the number of candidates per template node on several datasets from the MAA program. On some of these datasets, our filters find all of the signal nodes, and the signals can be counted directly. The number of signals can be quite large. For example, when there exist template nodes with very few (one or two) edges connecting them to the rest of the template, there tend to be many signals which differ only in the corresponding nodes.

For each node u in the template, we look for the set of nodes in the background that correspond to u in at least one signal. Initially, we consider every node in the background to be a candidate for each node in the template. We then eliminate candidates for nodes in the template by repeatedly applying filters until the sets of candidates converge. Next, we remove nodes from the background which are not candidates for any template node. Finally, we repeat these steps until no further nodes are removed from the background.

There are three possible outcomes:

- There are no candidates left for any node in the template. In this case, we can be sure that no signal exists in the background.

- The candidates converge to the nodes participating in signals, allowing us to enumerate the corresponding isomorphisms.
- The candidates do not converge to a small enough set, so we cannot easily determine how many signals exist – in this case a more sophisticated “counting” code would be required.

The goal of filtering is, for each template node u , to eliminate candidates in the background which cannot correspond to u in any signal.

A. Node-Level Statistics Filter

If a node v in the background belongs to a signal, it must correspond to some node u in the template. Then, certain statistical properties of the node v must be larger than the corresponding properties of u . For example, there must be at least as many edges incident to v as there are on u . Therefore, if there are fewer edges incident on v than there are on u , v can be eliminated as a candidate for u . Statistics which can be used to filter in this way include:

- In/out-degree
- Number of direct successors/predecessors
- Number of reciprocated edges
- Number of self-edges

Each of these statistics can be used separately for each channel in the networks.

B. Topology Filter

If nodes v_1 and v_2 in the background belong to a signal in which they correspond to nodes u_1 and u_2 in the template, there must be at least as many edges in each channel between v_1 and v_2 as there are between u_1 and u_2 . Therefore, if there are fewer edges in some channel between v_1 and every candidate for u_2 , v_1 can be eliminated as a candidate for u_1 . Likewise, if there are fewer edges in some channel between v_2 and every candidate for u_1 , v_2 can be eliminated as a candidate for u_2 .

C. Repeated Set Filter

This filter uses the fact that any isomorphism from the template to a signal must be injective. A consequence of this is that if there is a template node which has only one candidate, that candidate can be eliminated from candidacy for every other template node. More generally, if there is a set of k template nodes, the union of whose candidates is a set also of size k , then those candidates cannot be candidates for any template node outside the set. We relax this latter condition and instead impose that if a set of k template nodes each have exactly the same k candidates, those candidates are removed from the candidates of all other template nodes.

The repeated set filter is most important for templates which contain structures that remain invariant under some permutations of labels e.g. a template containing several leaf nodes, each of which is structurally interchangeable.

After applying the filters described above, we may find that most template nodes have exactly one candidate. However, some template nodes still have multiple candidates; we refer to these as *unspecified* nodes. To count the number of isomorphisms, we must count the number of ways that candidates can be chosen for these unspecified nodes. Each possible choice defines an isomorphism.

When an edge exists between two unspecified nodes, we have to enforce that the corresponding edge exists between the two candidates we choose for them. As this makes the problem of counting isomorphisms computationally complex, we instead find a set of unspecified nodes which, if specified, would cause the remaining unspecified nodes to have no edges between them. This set is called a *node cover*, and the smallest such set is called the *minimal node cover*. These methods are summarized in our 2021 IEEE Trans. Net. Sci. Eng. paper. We restate them here:

The overarching approach is a filtering method that is repeated until convergence.

Algorithm 1 Filtering

```

1: Input template  $\mathcal{G}_t$ , world  $\mathcal{G}_w$ , candidate set  $D(\mathbf{v}_t)$  for
   each  $\mathbf{v}_t \in \mathcal{V}_t$ , list of filters filters
2: converged  $\leftarrow$  False
3: while converged is False
4:   converged  $\leftarrow$  True  $\triangleright$  Stop unless progress is made
5:   for filter  $\in$  filters
6:      $D \leftarrow \text{filter}(\mathcal{G}_t, \mathcal{G}_w, D)$   $\triangleright$  Apply the filter
7:     if  $|D(\mathbf{v}_t)|$  decreased for some  $\mathbf{v}_t \in \mathcal{V}_t$ 
8:       converged  $\leftarrow$  False  $\triangleright$  Progress was made
9:       Break  $\triangleright$  Restart from the first filter
10: Output updated candidate set  $D(\mathbf{v}_t)$  for each  $\mathbf{v}_t \in \mathcal{V}_t$ 

```

The first filter is a node-level statistics filter. The idea behind the node-level statistics filter (Algorithm 2) is intuitive: for a world node $\mathbf{v}_w$ to be a candidate for a template node $\mathbf{v}_t$, certain statistical properties of $\mathbf{v}_w$ should not be less than those of $\mathbf{v}_t$. The idea of the node-level statistics filter has been applied to simpler settings in the related literature. Any statistic that is nondecreasing as nodes and/or edges are added to a graph can be used as part of the filter. The statistics that have been applied in our filter include in/out-degree, number of in/out-neighbors, number of reciprocated edges, and number of self-edges. Each of these statistics is used in each channel in the networks. Other more complex statistics can be used channel-wise, such as number of triangles, number of nodes within k steps, and number of paths of length k . Statistics combining information from multiple channels can also be used, such as the number of in-neighbors in channel a that are also out-neighbors in channel b . We use only simple statistics so that the cost of computing them scales only linearly in the number of distinguishable edges (edges whose source, destination, direction, and channel are distinct).

Algorithm 2 Node-level Statistics Filter

```
1: Input template  $\mathcal{G}_t$ , world  $\mathcal{G}_w$ , candidate set  $D(\mathbf{v}_t)$  for  
   each  $\mathbf{v}_t \in \mathcal{V}_t$   
2: for statistic  $\in [\text{in-degree}, \text{out-degree}, \dots]$   
3:   for template or world node  $\mathbf{v} \in \mathcal{V}_t \cup \mathcal{V}_w$   
4:     Compute statistic( $\mathbf{v}$ )  
5:   for template node  $\mathbf{v}_t \in \mathcal{V}_t$   
6:     for candidate  $\mathbf{v}_w \in D(\mathbf{v}_t)$   
7:       if statistic( $\mathbf{v}_w$ ) < statistic( $\mathbf{v}_t$ )  
8:         Remove  $\mathbf{v}_w$  from  $D(\mathbf{v}_t)$   
9: Output updated candidate set  $D(\mathbf{v}_t)$  for each  $\mathbf{v}_t \in \mathcal{V}_t$ 
```

The next filter is the topology filter that ensures that candidates have neighbors that are candidates for the neighboring nodes in the template.

Algorithm 3 Topology Filter

```
1: Input template  $\mathcal{G}_t$ , world  $\mathcal{G}_w$ , candidate set  $D(\mathbf{v}_t)$  for  
   each  $\mathbf{v}_t \in \mathcal{V}_t$   
2: for neighboring template nodes  $\mathbf{v}_t$  and  $\mathbf{u}_t$   
3:   for candidate  $\mathbf{v}_w \in D(\mathbf{v}_t)$   
4:     nbr_cand_found  $\leftarrow$  False  
5:     for candidate  $\mathbf{u}_w \in D(\mathbf{u}_t)$   
6:       enough_out  $\leftarrow \mathcal{E}_w(\mathbf{v}_w, \mathbf{u}_w) \geq \mathcal{E}_t(\mathbf{v}_t, \mathbf{u}_t)$   
7:       enough_in  $\leftarrow \mathcal{E}_w(\mathbf{u}_w, \mathbf{v}_w) \geq \mathcal{E}_t(\mathbf{u}_t, \mathbf{v}_t)$   
8:       if enough_out and enough_in  
9:         nbr_cand_found  $\leftarrow$  True  
10:      break  
11:   if nbr_cand_found is not True  
12:     Remove  $\mathbf{v}_w$  from  $D(\mathbf{v}_t)$   
13: Output updated candidate set  $D(\mathbf{v}_t)$  for each  $\mathbf{v}_t \in \mathcal{V}_t$ 
```

The next filter is the repeated-set filter. In some works this constraint is known as the AllDifferent constraint. This requires that for a world node $\mathbf{v}_w$ to be a candidate for template node $\mathbf{v}_t$, there must exist some injective mapping from the template nodes to their candidates under which $\mathbf{v}_t$ is mapped to $\mathbf{v}_w$. To enforce this constraint, we identify sets of template nodes T where the union of their candidates S has the same cardinality as T . These are known as tight sets. Candidates for template nodes in a tight set cannot be candidates for any nodes outside of the tight set, since this would violate the constraint. In the repeated-sets filter (Algorithm 4), we enforce the AllDifferent constraint by directly considering unions of candidate sets to find tight sets. Though the number of candidate set unions grows exponentially with the number of template nodes, we restrict this growth by not considering unions with more nodes than there are template nodes. We also keep track of template nodes that belong to tight sets and do not use

them in unions. In practice, template sizes are often small, and we don't observe the worst case exponential scaling. Our algorithm scales linearly in the number of world graph nodes.

Algorithm 4 Repeated-Sets Filter

```

1: Input template  $\mathcal{G}_t$ , world  $\mathcal{G}_w$ , candidate set  $D(\mathbf{v}_t)$  for
   each  $\mathbf{v}_t \in \mathcal{V}_t$ 
2:  $\text{unions} \leftarrow \text{EmptyMap}()$ 
    $\triangleright$  Map sets of world nodes  $\mathcal{U}_w$  to sets of template
       nodes  $\mathcal{U}_t$  for which  $\mathcal{U}_w = \bigcup_{\mathbf{v}_t \in \mathcal{U}_t} D(\mathbf{v}_t)$ . Any  $\mathcal{U}_t$  with
        $|\text{unions}[\mathcal{U}_t]| == |\mathcal{U}_t|$  is a tight set.
3:  $\mathcal{T} \leftarrow \emptyset$   $\triangleright$  Nodes known to belong to tight sets.
4:  $\text{todo} \leftarrow \{\mathbf{v}_t \in \mathcal{V}_t : |D(\mathbf{v}_t)| < |\mathcal{V}_t|\}$ 
    $\triangleright$  Template nodes with too many candidates cannot
       belong to a tight set.
5: while  $|\text{todo}| > 0$ 
6:    $\mathbf{v}_t \leftarrow$  element of  $\text{todo}$  with fewest candidates  $D(\mathbf{v}_t)$ 
7:    $\text{todo} \leftarrow \text{todo} \setminus \{\mathbf{v}_t\}$ 
8:   for  $(\mathcal{U}_w, \mathcal{U}_t) \in \text{unions}$ 
9:      $\mathcal{U}_w \leftarrow \mathcal{U}_w \cup D(\mathbf{v}_t)$ 
10:    if  $|\mathcal{U}_w| < |\mathcal{V}_t|$ 
11:       $\mathcal{U}_t \leftarrow \mathcal{U}_t \cup \{\mathbf{v}_t\}$ 
12:      if  $|\mathcal{U}_t| == |\mathcal{U}_w|$   $\triangleright \mathcal{U}_t$  is a tight set.
13:         $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{U}_t$ 
14:         $D(\mathbf{u}_t) \leftarrow D(\mathbf{u}_t) \setminus \mathcal{U}_w$  for  $\mathbf{u}_t \in \mathcal{V}_t \setminus \mathcal{U}_t$ 
15:      else
16:         $\text{unions}[\mathcal{U}_w] \leftarrow \mathcal{U}_t$ 
17:    if  $D(\mathbf{v}_t) \notin \text{unions}$ 
18:      if  $|D(\mathbf{v}_t)| == 1$   $\triangleright \{\mathbf{v}_t\}$  is a tight set.
19:         $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathbf{v}_t\}$ 
20:         $D(\mathbf{u}_t) \leftarrow D(\mathbf{u}_t) \setminus D(\mathbf{v}_t)$  for  $\mathbf{u}_t \in \mathcal{V}_t \setminus \{\mathbf{v}_t\}$ 
21:      else
22:         $\text{unions}[D(\mathbf{v}_t)] \leftarrow \{\mathbf{v}_t\}$ 
23:     $\text{todo} \leftarrow \{\mathbf{v}_t \in \text{todo} : |D(\mathbf{v}_t)| < |\mathcal{V}_t \setminus \mathcal{T}|\}$ 
24: Output updated candidate set  $D(\mathbf{v}_t)$  for  $\mathbf{v}_t \in \mathcal{V}_t$ 

```

The neighborhood filter extends the Local AllDifferent (LAD) constraint to the multiplex network case. In the undirected single-layer graph context, the LAD constraint ensures that for a world node $\mathbf{v}_w$ to be a candidate for a template node $\mathbf{v}_t$, there must be some injective mapping f from the neighbors of $\mathbf{v}_t$ to their candidates under which $f(\mathbf{u}_t)$ is a neighbor of $\mathbf{v}_w$ for each $\mathbf{u}_t$. We extend this to the multiplex network context by requiring not just that $f(\mathbf{u}_t)$ neighbors of $\mathbf{v}_w$, but also that there are enough edges in each channel and direction $E_w(\mathbf{v}_w, \mathbf{u}_w) \geq E_t(\mathbf{v}_t, \mathbf{u}_t)$ and $E_w(\mathbf{u}_w, \mathbf{v}_w) \geq E_t(\mathbf{u}_t, \mathbf{v}_t)$ (1) where $\mathbf{u}_w = f(\mathbf{u}_t)$. In the neighborhood filter (Algorithm 5), we enforce the multiplex LAD constraint by searching for such a mapping f . If none exists, we eliminate $\mathbf{v}_w$ as a candidate for $\mathbf{v}_t$.

Algorithm 5 Neighborhood Filter

```
1: Input template  $\mathcal{G}_t$ , world  $\mathcal{G}_w$ , candidate set  $D(\mathbf{v}_t)$  for  
   each  $\mathbf{v}_t \in \mathcal{V}_t$   
2: for template node  $\mathbf{v}_t \in \mathcal{V}_t$   
3:   for candidate  $\mathbf{v}_w \in D(\mathbf{v}_t)$   
4:      $\mathcal{N}_{\mathbf{v}_t} \leftarrow \text{Neighborhood}(\mathbf{v}_t)$   
5:      $\mathcal{N}_{\mathbf{v}_w} \leftarrow \text{Neighborhood}(\mathbf{v}_w)$   
6:      $\mathcal{B} \leftarrow \text{EmptyBipartiteGraph}(\mathcal{N}_{\mathbf{v}_t}, \mathcal{N}_{\mathbf{v}_w})$   
7:     for  $(\mathbf{u}_t, \mathbf{u}_w) \in \mathcal{N}_{\mathbf{v}_t} \times \mathcal{N}_{\mathbf{v}_w}$   
8:       if  $\begin{cases} \mathbf{u}_w \in D(\mathbf{u}_t) \\ \text{and } \mathcal{E}_w(\mathbf{v}_w, \mathbf{u}_w) \geq \mathcal{E}_t(\mathbf{v}_t, \mathbf{u}_t) \\ \text{and } \mathcal{E}_w(\mathbf{u}_w, \mathbf{v}_w) \geq \mathcal{E}_t(\mathbf{u}_t, \mathbf{v}_t) \end{cases}$   
9:         Add an edge between  $\mathbf{u}_t$  and  $\mathbf{u}_w$  in  $\mathcal{B}$   
10:     $\text{max\_match} = \text{MaxCardinalityMatching}(\mathcal{B})$   
11:    if  $|\text{max\_match}| < |\mathcal{N}_{\mathbf{v}_t}|$   
12:      Remove  $\mathbf{v}_w$  from  $D(\mathbf{v}_t)$   
13: Output updated candidate set  $D(\mathbf{v}_t)$  for each  $\mathbf{v}_t \in \mathcal{V}_t$ 
```

The elimination filter (Algorithm 6) eliminates candidates by identifying any contradictions that would result from them being assigned. For each template node candidate pair $(\mathbf{v}_t, \mathbf{v}_w)$, we do a one-step lookahead. We assign $\mathbf{v}_w$ to $\mathbf{v}_t$ and iterate over all other filters until convergence. If this results in one or more template nodes having no candidates, then $\mathbf{v}_t$ cannot be mapped to $\mathbf{v}_w$ and we eliminate $\mathbf{v}_w$ as a candidate for $\mathbf{v}_t$. As the elimination filter is a very expensive operation, scaling with the number of remaining candidates, we restrict its use until all other filters have converged and no further candidates can be removed by other means. In certain contexts, the elimination filter can be impractical to use. However, in others, it can greatly reduce the number of candidates.

Algorithm 6 Elimination Filter

```
1: Input template  $\mathcal{G}_t$ , world  $\mathcal{G}_w$ , candidate set  $D(\mathbf{v}_t)$  for  
   each  $\mathbf{v}_t \in \mathcal{V}_t$   
2:  $S \leftarrow \text{copy}(D)$  ▷ Save the candidate sets  
3: for template node  $\mathbf{v}_t \in \mathcal{V}_t$   
4:   for candidate  $\mathbf{v}_w \in D(\mathbf{v}_t)$   
5:      $D(\mathbf{v}_t) \leftarrow \{\mathbf{v}_w\}$  ▷ Assign  $\mathbf{v}_w$  to  $\mathbf{v}_t$   
6:     Iterate all filters to convergence  
7:     if  $D(\mathbf{u}_t)$  is empty for any  $\mathbf{u}_t \in \mathcal{V}_t$   
8:       Remove  $\mathbf{v}_w$  from  $D(\mathbf{v}_t)$  in  $S$   
9:     Reset candidate sets to  $S$   
10: Output updated candidate set  $D(\mathbf{v}_t)$  for each  $\mathbf{v}_t \in \mathcal{V}_t$ 
```

After applying the filters, some template nodes may have exactly one candidate. Template nodes that still have multiple candidates we refer to as unspecified nodes. When an edge exists between two unspecified nodes, we have to enforce that a corresponding edge exists between the two candidates we choose for them. As this makes counting isomorphisms computationally complex, we start by finding a set of unspecified nodes that, if specified, would cause the remaining unspecified nodes to have no edges between them. This set is called a node cover, and the smallest such set is called the minimal node cover. Next, we iterate through all possible choices of candidates for nodes in the node cover. For each choice, we reapply the topology and repeated-sets filters so we can be sure that any remaining candidates belong to signals. Since the remaining unspecified nodes have no edges between them, it is much simpler to count the ways to choose their candidates. The only constraint is that the same candidate cannot be chosen for more than one node, which is simply the AllDifferent constraint satisfaction problem.

Algorithm 7 Isomorphism Counting

```

1: function ISOCount
2:   Input template  $\mathcal{G}_t$ , world  $\mathcal{G}_w$ , candidate set  $D(\mathbf{v}_t)$  for
     each  $\mathbf{v}_t \in \mathcal{V}_t$ , unspecified node cover  $\mathcal{NC}$ 
3:   if  $\mathcal{NC} = \emptyset$ 
4:      $N_{\text{isos}} \leftarrow \text{CountAllDiff}(D)$ 
5:   else
6:      $N_{\text{isos}} \leftarrow 0$ 
7:      $\mathbf{v}_t \leftarrow \mathcal{NC}.\text{pop}()$ 
8:      $S \leftarrow \text{copy}(D)$   $\triangleright$  Save the candidate sets
9:     for world node  $\mathbf{v}_w \in D(\mathbf{v}_t)$ 
10:       $D(\mathbf{v}_t) \leftarrow \{\mathbf{v}_w\}$   $\triangleright$  Assign  $\mathbf{v}_w$  to  $\mathbf{v}_t$ 
11:      Apply filters
12:       $N_{\text{isos}} \leftarrow N_{\text{isos}} + \text{ISOCount}(\mathcal{G}_t, \mathcal{G}_w, D, \mathcal{NC})$ 
13:      Reset candidate sets to  $S$ 
14:   Output number of isomorphisms  $N_{\text{isos}}$ 

```

A simpler problem than listing or counting all isomorphisms (SMP or SICP) is the minimal candidate sets problem (MCSP). Here, we refer to the procedure for solving the MCSP as validation, in which we seek to identify all template node-candidate pairs that participate in at least one isomorphism. An algorithm for validation is as follows (Algorithm 8). First, pick an unvalidated pair of a template node and its candidate. Next, make assignments until an isomorphism is found, running filters after each step and backtracking as necessary. If all possible assignments fail, then the pair is invalid; remove the pair from the set of possible candidates. Otherwise, validate every assignment made as part of the isomorphism, including the initial one; these pairs have been validated as taking part in at least one isomorphism. Repeat this process until all pairs have been validated or removed from candidates. This ensures that every candidate-template node pairing remaining after validation is valid i.e. participates in at least one signal. This is because in order to pass validation, it must participate in at least one isomorphism that was found, and in order to fail, there must not exist an isomorphism containing it.

Algorithm 8 Validation

```
1: Input template  $\mathcal{G}_t$ , world  $\mathcal{G}_w$ , candidate set  $D(\mathbf{v}_t)$  for
   each  $\mathbf{v}_t \in \mathcal{V}_t$ 
2:  $D'(\mathbf{v}_t) \leftarrow \emptyset$  for each  $\mathbf{v}_t \in \mathcal{V}_t$   $\triangleright$  Valid candidates
3: for template node  $\mathbf{v}_t \in \mathcal{V}_t$ 
4:   for candidate  $\mathbf{v}_w \in D(\mathbf{v}_t) \setminus D'(\mathbf{v}_t)$ 
5:      $S \leftarrow \text{copy}(D)$ 
6:      $S(\mathbf{v}_t) \leftarrow \{\mathbf{v}_w\}$   $\triangleright$  Assign  $\mathbf{v}_w$  to  $\mathbf{v}_t$ 
7:     Attempt to find an SI  $f$  using  $S$   $\triangleright$  Solve the SIP
8:     if isomorphism found
9:        $D'(\mathbf{u}_t) \leftarrow D'(\mathbf{u}_t) \cup \{f(\mathbf{u}_t)\}$  for each  $\mathbf{u}_t \in \mathcal{V}_t$ 
10:    else
11:      Remove  $\mathbf{v}_w$  from  $D(\mathbf{v}_t)$ 
12: Output minimal candidate set  $D'(\mathbf{v}_t)$  for each  $\mathbf{v}_t \in \mathcal{V}_t$ 
```

Below we summarize the exact matching algorithm in a flowchart:

Exact Matching Algorithm Flowchart

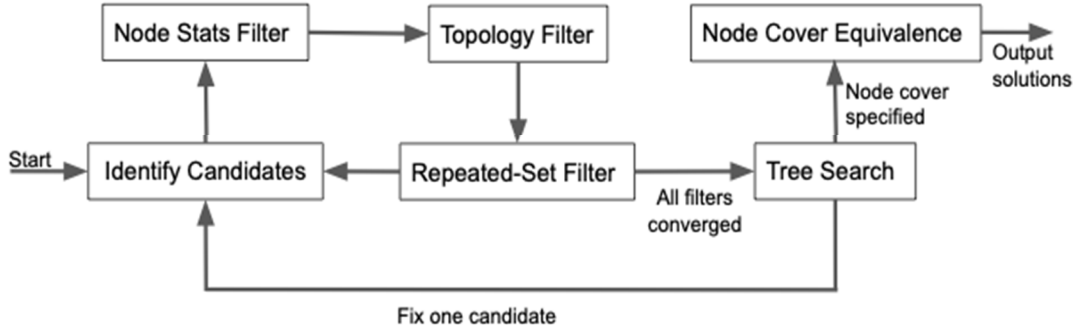


Figure 2: Exact matching algorithm flowchart

3.1.2 Phase II - Inexact subgraph matching on knowledge graphs

Phase II of the DARPA MAA program involved inexact subgraph matching on knowledge graphs. We introduce algorithms for inexact attributed subgraph isomorphism and matching to find optimal sub-graphs as measured by a graph edit distance. We show results for noisy queries on knowledge graphs. The graph edit distance is a measure of the distance between two graphs, as defined by the total cost of the cheapest sequence of edits which transform one graph into another. The six possible edits usually considered are node addition, node deletion, node substitution, edge addition, edge deletion, and edge substitution. In our work we considered a cost metric using only the latter four edits. This problem involves solving the global cost bound. Computing this bound is equivalent to solving a rectangular linear assignment problem (LAP) of

size $|V_t| \times |V_w|$. Many algorithms exist to solve the LAP and its rectangular variant. The most notable ones include the Hungarian algorithm, the Munkres algorithm, the JonkerVolgenant (JV) algorithm, and the auction algorithm. In our implementation, we apply a modified JV algorithm, designed to efficiently solve rectangular linear sum assignment problems via a shortest augmenting path approach.

When data is attributed, we can not rely on exact matching logic. We do not want to disqualify matches for small misspellings, 1% difference in numerical value, etc. The new problem is to find “close” matches, as defined by a cost metric. We want to find the minimum cost subgraph or the top k subgraphs. The cost is expressed in terms of a graph edit distance from the template. Missing edges are allowed, but have a cost penalty. Mismatched attributes are also penalized. The cost per attribute depends on the attribute distance. We use flat costs for missing attributes and impose no penalty for having extra attributes.

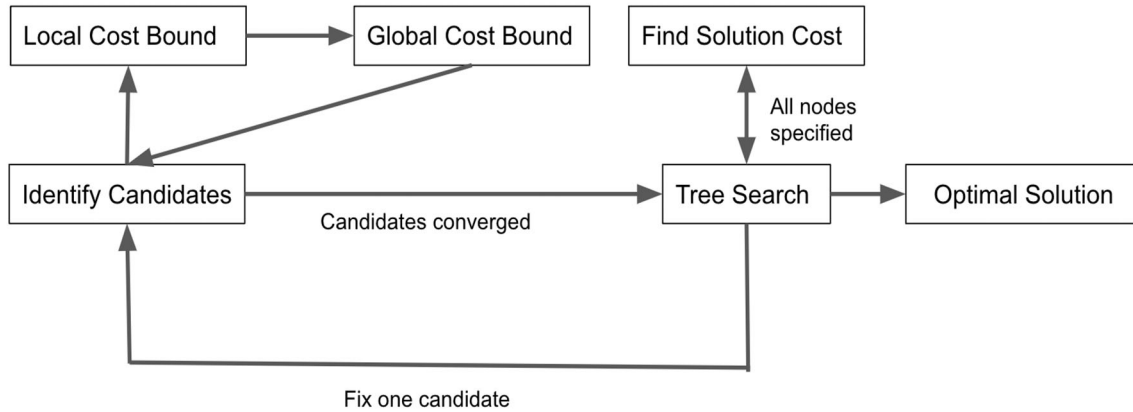


Figure 3: Flowchart for inexact subgraph matching code

3.2 Task 2

Our method from Phase I can be proved to solve the noiseless TA3 problem completely. Our algorithm was used by the GORDIAN team as a benchmark for their theoretical estimate on the number of subgraph isomorphisms. Qualitatively the results compare well (e.g. they are monotonic increasing on the datasets).

3.2.1 Phase I Structural Equivalence in subgraph matching

Our work on structural equivalence started in Phase I with (Nguyen et al 2019 IEEE BIG DATA). We have further developed this work into a journal paper currently under review (Yang et al 2022). Here is the basic definition of structural equivalence:

Definition 3. 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) $(v, w) \in E \Leftrightarrow (w, v) \in E$

Here are two propositions that are proved in this paper:

Proposition 6. Given graphs $G_T = (V_T, E_T)$, $G_W = (V_W, E_W)$, if $v, w \in V_T$ are structurally equivalent, then they are interchangeable in any subgraph isomorphism.

and

Proposition 8. If $v', w' \in V_W$ are structurally equivalent, then in any subgraph isomorphism f , they are interchangeable.

Together they can be used to prove the following proposition about the combinatorial complexity of the solution space given certain structural equivalences.

Proposition 9. Let $f : V_T \rightarrow V_W$ be a subgraph isomorphism. Suppose we partition the template graph $V_T = \bigcup_{i=1}^n C_i$ and the world graph $V_W = \bigcup_{j=1}^m D_j$ into structural equivalence classes. Let $C_{i,j} = C_i \cap f^{-1}(D_j)$ represent the set of template vertices in C_i that map to world vertices in the equivalence class D_j . Then there are

$$\prod_{i=1}^n |C_i|! \prod_{j=1}^m \prod_{i=1}^n \binom{|D_j| - \sum_{k=1}^{j-1} |C_{k,j}|}{|C_{i,j}|}$$

isomorphisms generated by interchanging equivalent template vertices or world vertices using Propositions 6 and 8.

Equivalence Hierarchy: The equivalence discussed in the prior section is a static form of equivalence, only taking into account information provided at the start of the subgraph search. However, as a subgraph search proceeds, we may be able to discard additional nodes and edges based on information derived from the assignments already made.

We developed a three other definitions of equivalence known as “candidate equivalence”, “node cover equivalence”, and “full equivalence”.

We develop algorithms to utilize these notions and have results presented in the next section.

3.2.2 Phase II Fault-tolerant Subgraph Matching on Aligned Networks

In the context of networks curated from real world data, an important problem is that of aligning entities across multiple data sources. Methods for performing alignment are often error-prone due to lack of information and noise within the data itself. When performing subgraph matching on the resulting aligned networks, these alignment errors often lead to catastrophic results. We developed a fault-tolerant algorithm for subgraph matching on aligned networks that takes into account potential alignment errors.

When dealing with data involving relationships between entities, it is often natural to store information in the format of a graph or a network. This allows for various graph analysis techniques to be applied to the data, such as community detection, centrality metrics, and subgraph matching, the last of which we focus on in this paper. Often, we wish to combine multiple graphs together in order to better express the relationships between their nodes. For example, different datasets representing multiple modes of communication between the same set of entities can be combined into a single multilayer network. However, in these situations, the correspondence between nodes in different datasets is sometimes unknown; though two nodes in different datasets may correspond to the same real world entity, the knowledge of which nodes correspond to which is often unknown. The problem of identifying this correspondence is the graph alignment problem (also known as the graph matching problem or network alignment problem) and has been well studied in the literature. Separate from this is the problem of subgraph matching: finding a subgraph which is a match for a given pattern or template. This match can take many forms, the most common and well studied of them being structural i.e. the subgraph isomorphism, which preserves edges between nodes. Simple extensions of this concept can include attributes or labels on the nodes and edges, which must also be matched between the template and the subgraph. However, in the context of aligned graphs, existing subgraph matching methods will often fail if there are errors in the alignment. Though it would be desirable to have an aligned graph without errors to perform subgraph matching on, this is often impractical or impossible to obtain in a real-world context.

We define an alignment on a set of networks as a set of disjoint sets of nodes contained within those networks. Each of these sets is referred to as an individual alignment, or sometimes just alignment when the context is clear. Given a set of networks $G_1, G_2, \dots$ and an alignment A , construct the aligned network G aligned by replacing each set of nodes in the alignment with a single new node whose edges are the edges of the nodes in the set. We define a minimal alignable subgraph isomorphism is an alignable subgraph isomorphism where no candidate can be removed without invalidating the conditions for alignable subgraph isomorphism.

We have proved the following:

Theorem 1

Given multiple graphs $\{G_1, G_2, \dots\}$, their ground truth alignment AGT , and any other alignment with no incorrect individual alignments A , every subgraph isomorphism on the ground truth corresponds to at least one minimal alignable subgraph isomorphism on the aligned graph.

The proof can be found at the end of (Tu and Yang 2021).

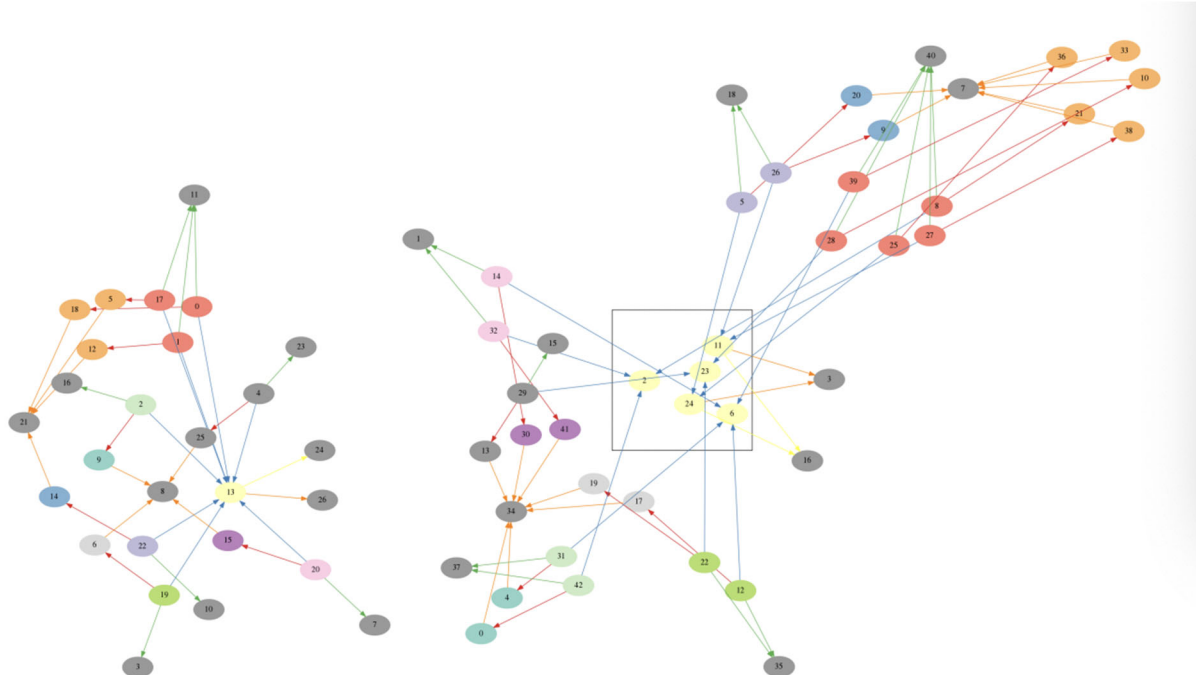


Figure 4 AIDA V2.0.2 template graph(left) and subgraph of world graph with nodes participating in minimal alignable subgraph isomorphism(right).

Figure 4 shows AIDA V2.0.2 template graph(left) and subgraph of world graph with nodes participating in minimal alignable subgraph isomorphism(right). Groups of nodes in the world graph of one color (that is not dark gray) are the candidates for template nodes of the same color. The boxed group in the world graph are the nodes that are involved in a merging.

3.3 Task 3

The representative solutions found during subgraph search and the associated solution classes they generate are essential to producing a compact visual description of the solution space. We have developed a tool to image these spaces via expressive graphical representations. If we use template structural equivalence, equivalent template vertices are interchangeable. We indicate this by pairing template equivalence classes with world vertices; producing a solution then amounts to picking a unique representative from each class. If we use candidate or node cover equivalence, equivalence classes are recomputed before each assignment. Because of this, we can have the peculiar case where a template node is paired with a world equivalence class that has been previously assigned but has grown in size due to recomputing equivalence. The subgraph of all nodes and edges that participate in the representative solution is the solution-induced world subgraph. The final graph depicts the compressed solution-induced world subgraph where we drop all nonparticipant nodes and edges, and we combine like-colored nodes into “supernodes” with a label indicating the number of nodes joined. From this last graph, we can observe the original template graph structure among the participant world nodes. We use these graphical representations to make apparent the various symmetric features of our datasets.

As an example, we plot the template and world subgraph for an example from biochemical dataset in the figure below.

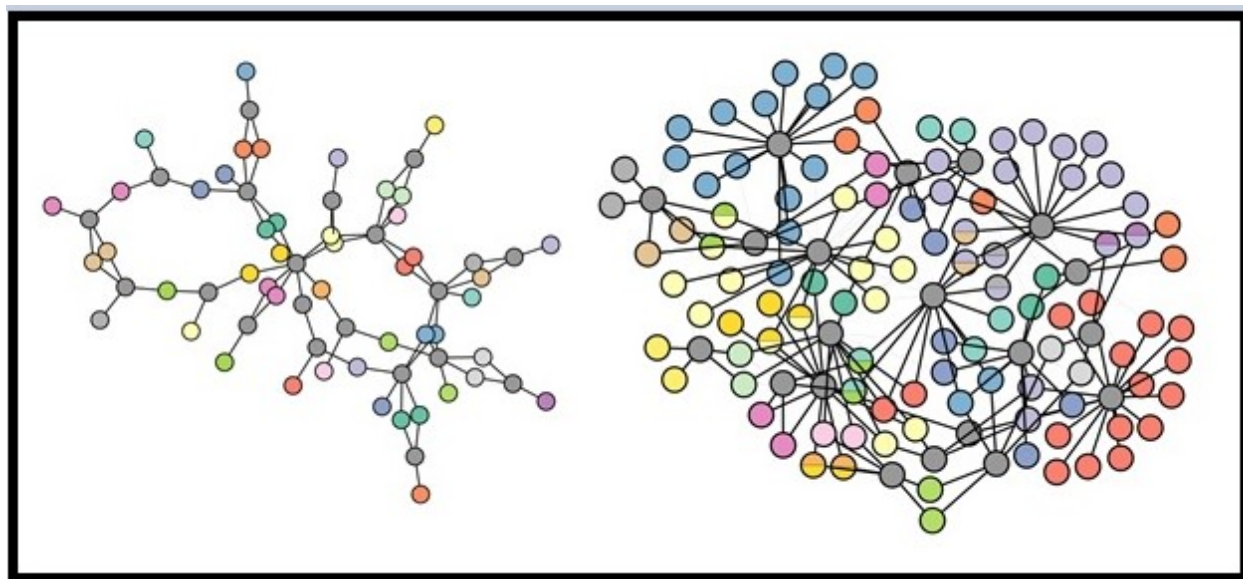


Figure 5: A biochemical reactions template graph (left) and the solution-induced world subgraph (right) for a solution class comprised of 9.18×10^{13} solutions.

Dark gray nodes are nodes with a single candidate. Nodes with the same non-gray color in the world subgraph are fully candidate equivalent. Nodes with two or more colors were part of one class at an early stage of subgraph search which was later merged into another class. All solutions represented by the compressed solution can be generated by mapping templates nodes of one color to world nodes with the same color.

From this depiction, we observe that there are multiple different sources from which equivalence may arise. One aspect is the large number of pairs of structurally equivalent nodes that are colored the same in the template graph. A second source is leaf nodes on the template graph that can be mapped to large equivalence classes in the world graph. By using an equivalence-informed subgraph search, we can expose exactly where these complexities arise, and an analyst could specify constraints to produce the exact solution desired. We note that there are further sources of equivalence not addressed by this representation, for example automorphic equivalence (observe the substructures of 5 nodes which are repeated on the perimeter of the template graph).

4.0 RESULTS AND DISCUSSION

4.1 Phase I results

Here are the results of our filtering methods applied to the DARPA MAA datasets from Phase I of the MAA program. It is important to note that our results differ from other TA3 performers in a significant way. Rather than finding one isomorphism, we tackled the hard problem of understanding the entire solution space.

4.1.1 Constraint Propagation Code

We found that the problems posted in Phase I had solution spaces that were combinatorially complex and this led to some new work in this research area. These results are published in (Moorman et al 2021, IEEE TSNE).

Table 1. Results of our exact matching code for the Phase I DARPA MAA datasets.

14

Dataset	Instance	Template		World		Channels	Number of Isomorphisms	Filters	Problems Solved
		Nodes	Edges	Nodes	Edges				
PNNL Version 6	B0-S0	74	1,620	22,996	12,318,861	7	1,152	S, T, R	SIP, SNSP, MCSP, SICP, SMP
	B5-S0	64	1,201	22,994	12,324,975	7	1,152	S, T, R	SIP, SNSP, MCSP, SICP, SMP
	B1-S1	75	1,335	22,982	12,324,340	7	1,152	S, T, R, E	SIP, SNSP, MCSP, SICP, SMP
	B7-S1	81	1,373	23,011	12,327,168	7	3.13×10^8	S, T, R, E	SIP, SNSP, MCSP, SICP
PNNL Real World		35	158	6,407	74,862	3	2.12×10^{12}	S, T, R, N, E	SIP, SNSP, MCSP, SICP
GORDIAN Version 7	Batch-1	156	3,045	190,869	123,267,100	10	4.27×10^{15}	S, T, R, E	SIP, SNSP, MCSP, SICP
	Batch-2	44	715	190,869	123,264,754	10	1.35×10^{16}	S, T, R, E	SIP, SNSP, MCSP, SICP
IvySys Version 7		92	195	2,488	5,470,970	3	N/A	S, T, R, N, E	SIP
IvySys Version 11		103	387	1,404	5,719,030	5	N/A	S, T, R, E	SIP

For each instance of the datasets, the table records its basic statistics, which filters have been applied, and the number of isomorphisms. The last column concludes the types of problems stated in Section 1.1 that we can solve for each instance. The names of the filters have been abbreviated: L = Node Label; S = Node-level Statistics; T = Topology; R = Repeated-Sets; N = Neighborhood; E = Elimination.

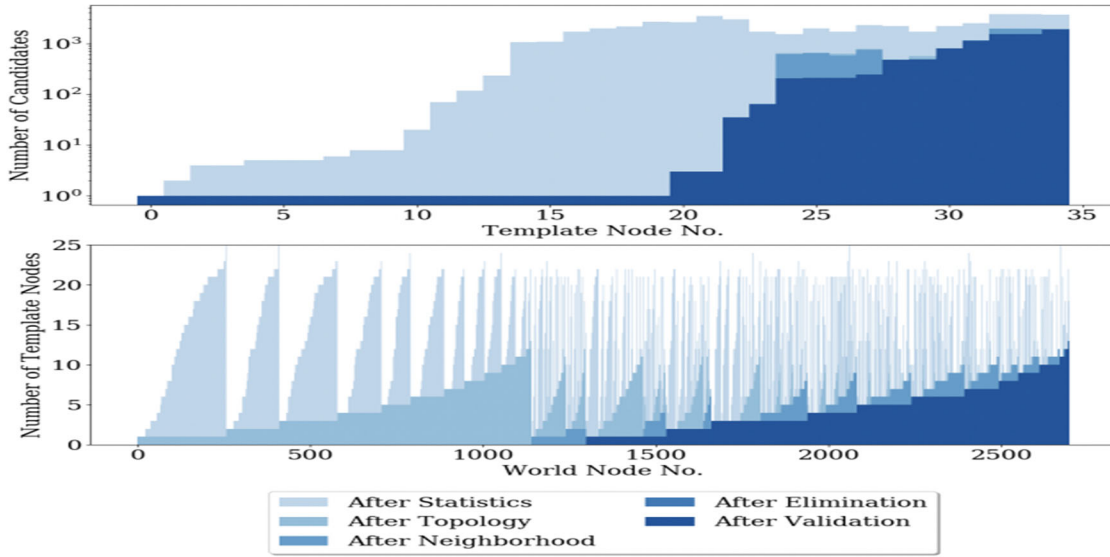


Figure 6: Number of candidates for each template node(top) and number of template nodes for which each world node is a candidate(bottom)

Figure 6 shows some details for the PNNL “Real World” dataset because it was the “challenge” problem at the end of Phase I. The figure on the right shows the number of candidates for each template node after different levels of filtering are applied to PNNL Real World (top). The bottom panel shows the number of template nodes for which each world node is a candidate. Note that the Validation histogram almost perfectly overlaps the Elimination histogram. We were able to find all the isomorphisms for this dataset - over 2×10^{12}

We also considered some real world examples of multichannel datasets that were not part of the DARPA MAA program, for comparison. Here are the results of our filtering algorithms on those datasets.

Dataset	Template		World		Channels	Number of isomorphisms	Filters	Problems solved
	Nodes	Edges	Nodes	Edges				
Britain Transportation	53	56	262,377	475,502	5	N/A	S, T, R, E	SIP
Britain Transportation (3km)	53	56	262,377	475,502	5	3.76×10^7	L, S, T, R, E	SIP, SNSP, MCSP, SSCP
Higgs Twitter	115	2,668	456,626	5,367,315	4	1.03×10^{14}	S, T, R	SIP, SNSP, MCSP, SSCP
Commercial Airlines	37	210	450	7,177	37	3.65×10^9	S, T, R	SIP, SNSP, MCSP, SSCP

4.1.2 Structural Equivalence

We have also developed a theory of structural equivalence that leads to a reduction in complexity of the understanding of the solution space. The first set of datasets we worked with on structural equivalence are a benchmark dataset for subgraph matching on single channel networks.

Table 2. Benchmark datasets for single channel subgraph match study.

BENCHMARK DATASET STATISTICS													
Dataset	# Instances	# Nodes		Template # Edges		Density		# Nodes		World # Edges		Density	
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max
SF	100	180	900	478	5978	0.006	0.165	200	1000	592	7148	0.006	0.159
LV	6105	10	6671	10	209000	0.001	1.000	10	6671	10	209000	0.001	1.000
SI	1170	40	777	41	12410	0.005	0.209	200	1296	299	34210	0.004	0.191
images-cv	6278	15	151	20	215	0.019	0.190	1072	5972	1540	8891	0.000	0.003
meshes-cv	3018	40	199	114	539	0.022	0.146	201	5873	252	15292	0.000	0.022
images-pr	24	4	170	4	241	0.017	0.667	4838	4838	7067	7067	0.001	0.001
biochemical	9180	9	386	8	886	0.012	0.423	9	386	8	886	0.012	0.423
phase	200	30	30	128	387	0.294	0.890	150	150	4132	8740	0.370	0.782

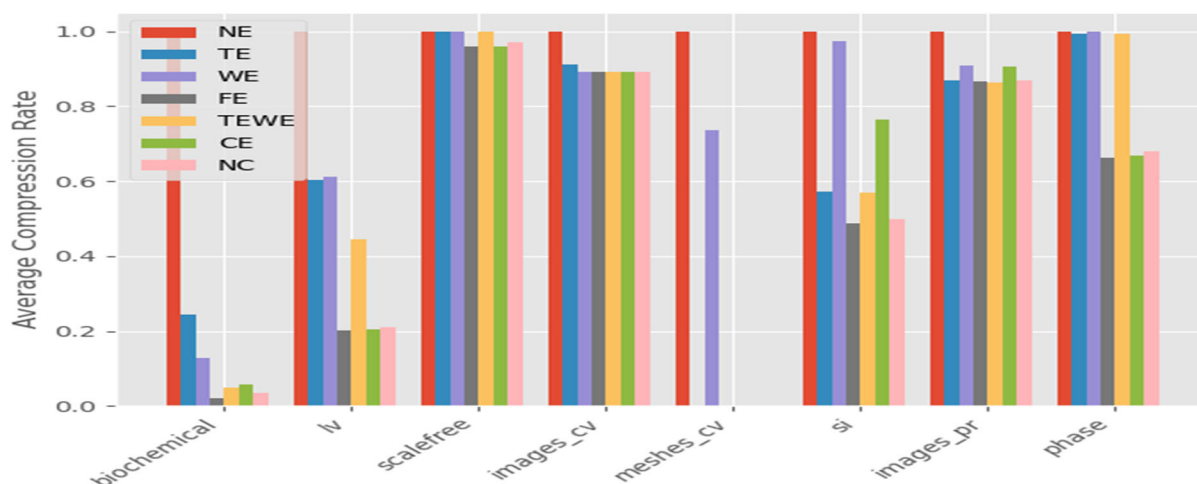


Figure 7: a diagram for the average compression rate for the single channel benchmark networks above using different algorithms for each notion of equivalence

In Figure 7 above, we have a diagram for the average compression rate for the single channel benchmark networks above using different algorithms for each notion of equivalence. Compression rate is the ratio of the number of isomorphisms found with no equivalence divided by the number found with each type of equivalence. The trials were each given 600 seconds to complete the task. The abbreviations are for NE = No equivalence; TE= template equivalence; WE = world equivalence; FE = full equivalence; TEWE= template and world structural equivalence; CE= candidate equivalence; NC = node cover equivalence.

Table 3: multichannel datasets on which we tested structural equivalence algorithms with the results on the right.

Dataset	Template		World		Channels
	Nodes	Edges	Nodes	Edges	
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

Table 4: Run times to find all the subgraph isomorphisms using different notions of equivalence.

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

Above we have Table 3 on the left of the multichannel datasets on which we tested structural equivalence algorithms with the results on the right. We stopped the code after 3600 seconds. Table 4 (right) are the run times to find all the subgraph isomorphisms using different notions of equivalence. The abbreviations are for NE = No equivalence; TE= template equivalence; WE = world equivalence; FE = full equivalence; TEWE= template and world structural equivalence; CE= candidate equivalence; NC = node cover equivalence. The PNNL, GORDIAN, and IvySys datasets were all created by TA1 performers for the DARPA MAA program.

4.2 Phase II results

We have a number of Phase II results. All of these datasets were complex knowledge graphs with attributes. In some cases we ran inexact matching code and in other cases we ran the exact match code.

4.2.1 AIDA Dataset v 2.1.2

First we report on results for the AIDA dataset, published in (Tu et al 2020) on inexact subgraph matching.

We test our algorithms on the AIDA Version 2.1.2 dataset created by Pacific Northwest National Laboratory (PNNL) for the DARPA-MAA program. This dataset consists of a knowledge graph collected from news articles about political events in the Ukraine. Provided also is a measure of distance between attributes, which can be used to construct the corresponding graph edit distance. The world graph has 98,817 nodes and 138,127 edges. Three templates are provided, each with six different variations labeled A-F. These templates are much smaller than the world graph, consisting of 11-33 nodes and 11-40 edges. These templates were created by taking a known “ground truth” existing subgraph and adding increasing levels of noise. The A template has no

noise and is guaranteed to have at least one exact match. Provided also is the ground truth mapping for the A version of each template. All experiments were run on an HP Z8 G4 Workstation with two 12-core 6136 3.0 2666MHz CPUs.

We perform top-k inexact approximate subgraph matching on these templates, with $k = 5$. We use two approaches: the first approach (labeled “Normal”) in the table, simply runs the algorithm with no initial cost threshold, while the second uses the ground truth cost bound (labeled “GTCB”) from the A template to derive an appropriate initial cost threshold for the other templates in each series (1,2,3). To identify this threshold, we first impose the matching from the ground truth, then set our algorithm to find optimal assignments for any remaining nodes that were not included in the ground truth. Using only this initial cost threshold, we discard all other matching information from the ground truth and proceed to optimize over the space of matches with lower cost than the cost of the ground truth. The cost of the ground truth for each template is listed in the Ground Truth column under Cost. In real world contexts, the ground truth is unknown. However, it is not unreasonable that another, possibly suboptimal, approximate match exists that we can use for the purpose of setting the initial cost threshold to restricting the solution space. The results of the two approaches on the AIDA Version 2.1.2 dataset are shown in the table below. Although we perform top-k matching, we list only one cost for each template; this is because for all templates considered, all 5 matches found were of the same cost. We observe that for all of the A templates, both the normal and GTCB approaches converge to the expected result of an exact match with 0 graph edit distance. For all templates other than 1C-F, both approaches were able to complete within the runtime limit of 46.5 hours, and arrive at the optimal solutions. For 1C-F, the normal version was aborted due to the runtime limit, resulting in the best solutions found during that time limit. For the GTCB version, only template 1E was aborted early, after 35.9 hours. This was done because, when examining the branching structure of the search algorithm, the approach was deemed unlikely to complete. The other three templates completed and reached optimal solutions.

Table 5: Results from out inexact matching code for AIDA v2.1.2.

Template	Time		Cost		
	Normal	GTCB	Normal	Ground Truth	GTCB
1A	27.8 min	1.5 sec	0.0	0.0	0.0
1B	24.5 min	0.2 sec	0.347	0.351	0.347
1C	46.5 hrs	5.48 hrs	8.783	3.610	2.254
1D	46.5 hrs	64.1 min	15.470	1.639	1.636
1E	46.5 hrs	35.9 hrs	16.194	2.642	2.638
1F	46.5 hrs	76.7 min	13.596	2.328	1.772
2A	2.25 min	0.06 sec	0.0	0.0	0.0
2B	3.07 min	0.15 sec	0.307	0.335	0.307
2C	6.35 min	10.6 sec	4.070	4.891	4.070
2D	12.4 min	32.9 sec	3.839	5.942	3.839
2E	4.58 min	22.4 sec	3.848	4.533	3.848
2F	18.5 min	42.3 sec	4.012	4.704	4.012
3A	2.33 min	0.08 sec	0.0	0.0	0.0
3B	2.27 sec	0.15 sec	0.145	0.191	0.145
3C	4.10 min	3.56 sec	2.452	2.452	2.452
3D	10.3 min	2.48 min	2.312	2.585	2.312
3E	20.8 min	10.9 min	2.472	3.471	2.472
3F	4.80 min	0.94 sec	1.314	1.348	1.314

Table 5 shows the 17 templates for this challenge problem. Run time for each template is shown for both the normal version of our code and the ground truth cost bound version of our code. Likewise we show the minimum cost achieved for the inexact match.

Table 6: Aida v. 2.1.2 from Phase II of the MAA program.

	Template					
	1A	1B	1C	1D	1E	1F
# Solutions	6	6120	324	>392k	>382k	>400k
# Rep. Sols.	3	2520	162	>315k	>34k	>58k
# World Nodes	38	99	47	109	3169	3141
# Eq. Classes	37	95	46	105	1686	1692
	2A	2B	2C	2D	2E	2F
# Solutions	78	1400	39	13780	1248	17368
# Rep. Sols.	6	60	3	1160	128	4160
# World Nodes	29	41	27	45	37	48
# Eq. Classes	17	23	15	30	26	36
	3A	3B	3C	3D	3E	
# Solutions	6	6	36	198	198	
# Rep. Sols.	4	4	16	92	92	
# World Nodes	16	16	21	26	23	
# Eq. Classes	15	15	18	23	23	

After identifying optimal solutions (for all templates except 1E), we then apply our approach for inexact ASM to find all optimal solutions by setting the cost threshold to the cost of an optimal solution. The number of optimal solutions found is listed in the table below. In the case of 1E, we set this cost threshold to be the cost of the best solution found; we observe that all found solutions were of the same cost. For templates 1D, 1E, and 1F, the optimal solution search was cut off after 209 hours due to time and memory limitations. We believe that there exist more solutions than those that were found, possibly orders of magnitude more. We have marked these templates with a > to indicate this.

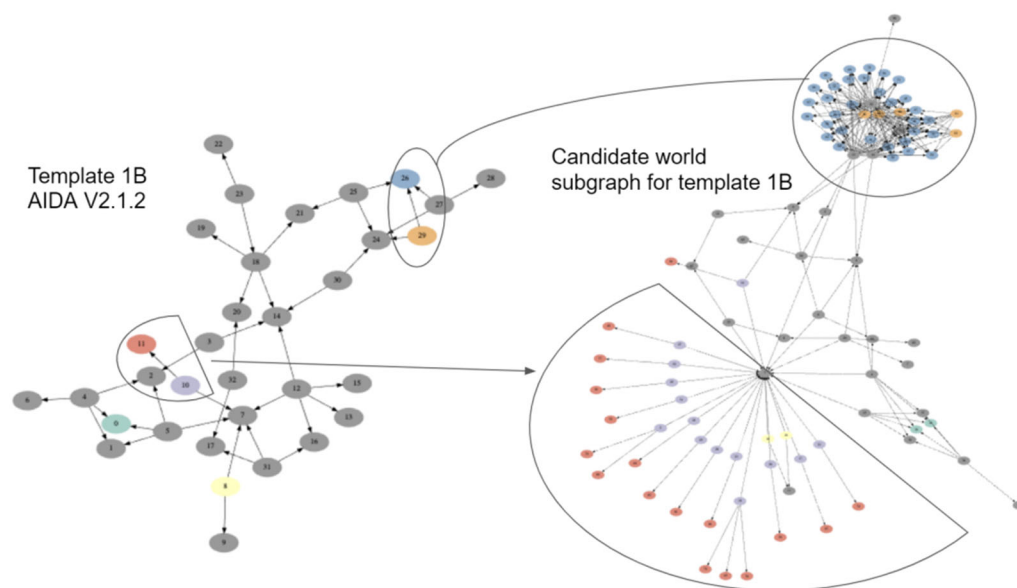
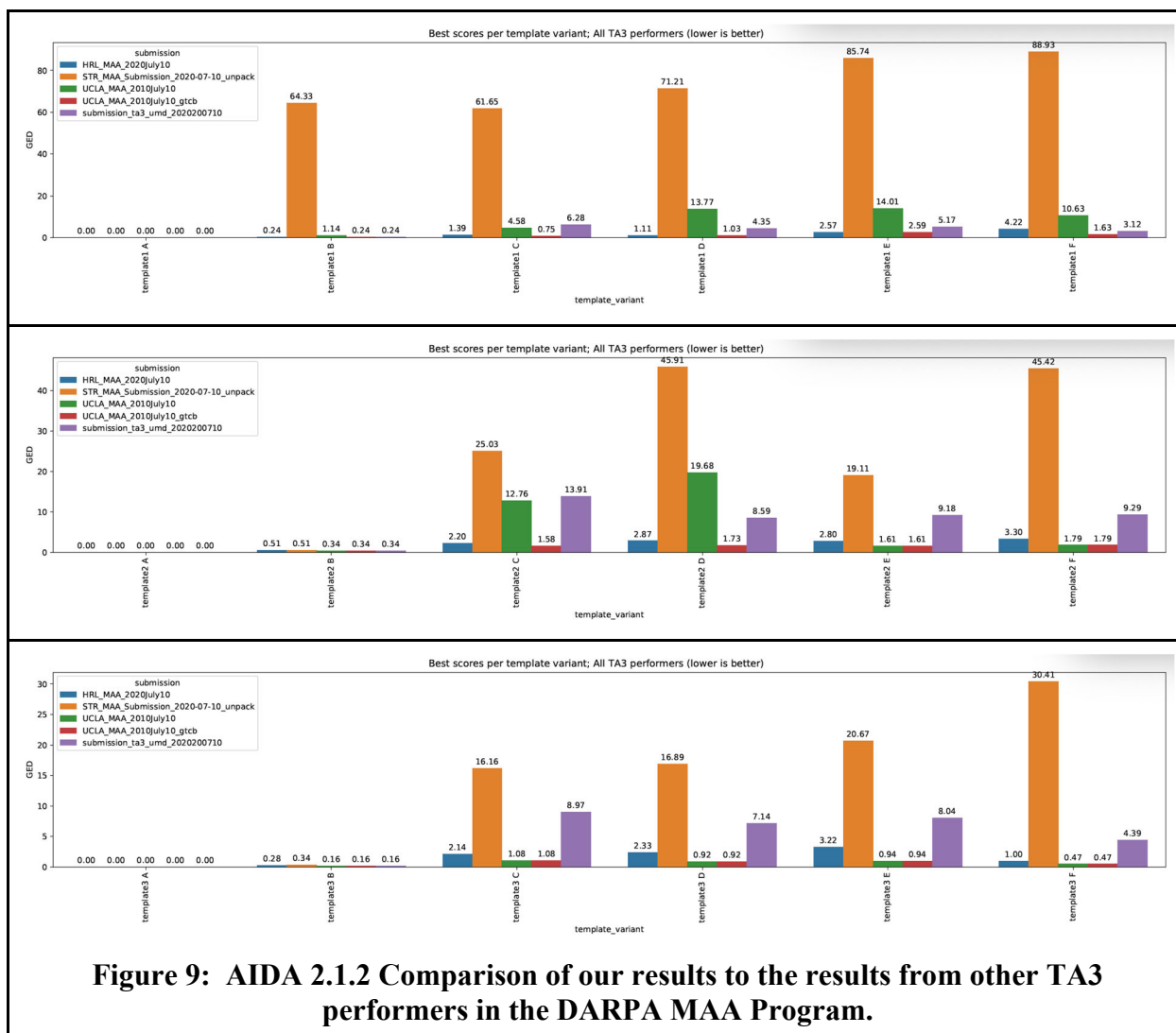


Figure 8: Diagram of the solution space for Template 1B for the AIDA v2.1.2 dataset.



We have the top performing code for this dataset. Lower cost is good. Graph edit distance cost is shown on the vertical axis. UCLA has two versions of code. The normal version assumes nothing about the result. The GTCB has knowledge of the “ground truth cost bound” but not the actual solution. This data was provided by Pacific Northwest National Laboratory.

Table 7: # of solutions found for each template of AIDA 2.1.2.

Template	# Solns	# Comp. Solns.	# World nodes	# W. Equiv. Cls.
1A	6	3	38	37
1B	6120	2520	99	95
1C	324	162	47	46
1D	392091	315336	109	105
1E	382222	34314	3169	1686
1F	400457	58213	3141	1692
2A	78	6	29	17
2B	1400	60	41	23
2C	39	3	27	15
2D	13780	1160	45	30
2E	1248	128	37	26
2F	17368	4160	48	36
3A	6	4	16	15
3B	6	4	16	15
3C	36	16	21	18
3D	198	92	26	23
3E	198	92	23	23

Table 7 is a chart with the number of solutions found for each template of AIDA 2.1.2. In the second column are the number of compressed solutions. In the third column are the number of world nodes and the fourth column are the number of world equivalence classes.

4.2.2 COVID Paths Dataset

Many mechanisms within biological systems can be modeled as pathways, chains of interactions between proteins, genes, chemicals, and other biological entities. These interactions can be represented using a graph structure, more specifically a knowledge graph representing known or inferred information about the entities in question. In this context, we propose a constraint propagation approach for identifying paths in a graph structure which represent potential biological pathways. We apply this approach to a knowledge graph dataset which was semantically extracted from literature on COVID-19. These results were published in (Tu 2021 - IEEE BIG DATA 2021).

A knowledge graph dataset on Coronavirus Disease 2019 (COVID-19) was created by Pacific Northwest National Laboratory (PNNL) as part of the DARPA-MAA program. Per PNNL's technical description, this dataset represents causal knowledge about biological entities which was automatically extracted from the COVID-19 Open Research Dataset using the INDRA knowledge assembler, the Blender knowledge graph, and the Comparative Toxicogenomics Database. Within this context, we seek to identify biological activation pathways that answer questions about COVID-19, such as how Severe Acute Respiratory Syndrome Coronavirus 2 (SARSCOV-2) induces a cytokine storm in severely ill COVID19 patients. To identify such pathways, we first model our biological query as a path-based graph template with constraints. We then construct a constraint propagation scheme taking into account the variable length of the path as well as the specified constraints. Finally, we perform a depth-first search for candidate paths that match the template. Though we focus on biological knowledge graphs in this paper, our approach also has applications to other fields, such as adversarial activity detection. In the problem context posed by DARPA-MAA, activities are represented in the form of graphs, and subgraph matching is used to identify matches for a template describing potential adversarial activity. However, certain types of adversarial activity, such as chains of shipments moving hazardous materials from one location to another, can be better modeled as variable length constrained paths rather than as a fixed set of nodes and edges. In the future, we seek to incorporate this pathfinding approach into a more flexible subgraph matching routine for templates that contain constrained paths.

4.2.2A Version 3.2.1

For version 3.2.1, the world graph has 85,938 nodes and 3,138,683 edges. There are four types of nodes: 13,377 nodes of type "Chemical", 2634 "Disease", 46445 "Gene", and 23,482 "Protein". The edges are more varied, being broken down into 184 types.

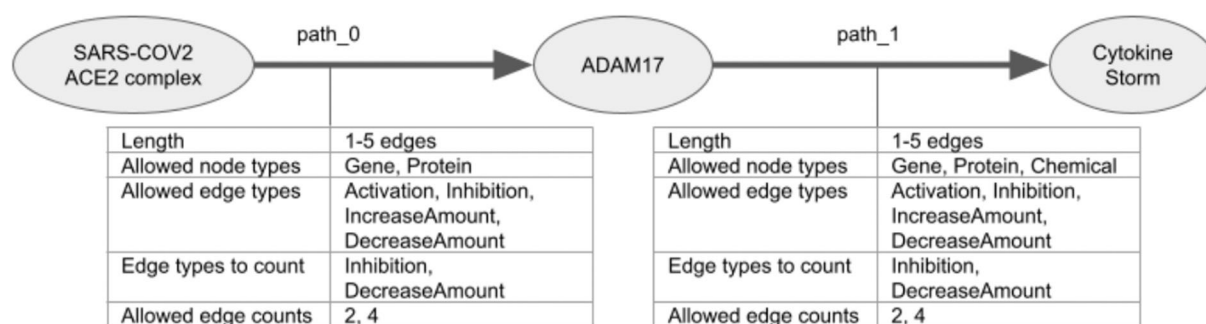


Figure 10: Template 1 is a structural template describing IL-6 amplification, a proposed mechanism for how COVID-19 induces cytokine storms.

This template takes the form of two paths. The first path is an activation pathway from two possible nodes representing the SARS-CoV-2 ACE2 complex to ADAM metallopeptidase domain 17/10 (ADAM17), and is constrained to have between one and five edges (inclusive), with either two or four edges being of type “Inhibition” or “DecreaseAmount”. The nodes in the first path are constrained to be of type “Gene” or “Protein”. The second path is another activation pathway from ADAM17 to various representations of cytokine storm in the dataset; it is also constrained to have between one and five edges, with two or four inhibitive edges. However, the nodes in the second path, in addition to “Gene” and “Protein”, are also allowed to be of type “Chemical”. Note that the template does not allow a count of zero inhibitive edges, despite zero being a valid option for an activation pathway; this was later noted by PNNL to be an issue with the provided template, but was not fixed for this version of the dataset. To cut down on the number of possible paths, as well as to ensure that only activation relationships are represented in the path, we imposed an additional constraint that did not appear in the template, namely that activating edges must be of type “Activation” or “IncreaseAmount” to parallel the inhibiting edge types of “Inhibition” and “DecreaseAmount”. The original template did not place any constraint on which types could be considered activating, except that they not be “Inhibition” or “DecreaseAmount”. Thus, initial attempts often included paths with non-activation edge types treated as activation, such as “affects^binding”, “affects^cotreatment”, or “decreases^reaction”. We find 5,591,288 matches for the first path constraint in the template and 13,529,145 matches for the second path constraint. Although this solution space is very large, it results from a surprisingly small number of candidates: the largest number of candidates for a given template role is only on the order of a few thousand. The huge number of solutions is thus not due to a large number of disjoint matches, but rather a large number of overlapping matches. We also observe that the largest number of candidates appear in the middle of the path, in positions 2 and 3. This makes sense, as the ends of the path are more constrained, being fixed to one of only a few possible values. Due to the enormity of the solution space, it is unlikely that all results for this template provide reasonable alternatives for explaining the link between SARS-COV-2 and cytokine storm formation. However, the relatively small size of the candidate set gives hope that, with some additional constraints on the path, a concise representation of the solution space can be identified and presented to a domain expert for further investigation.

4.2.2B Version 3.3.0

Version 3.3.0 of the COVID dataset has approximately 140 thousand nodes and 3.5 million edges. The dataset comes with 1,874 templates, each one modeling a path from a different drug to SARS-CoV-2. These paths model inhibition pathways, and thus are constrained to have an odd number of inhibitive edges, rather than an even number. A plot of this template is shown in Figure 5. This dataset also includes a comprehensive list of edge types, with each type labeled as activating (+1), inhibiting (-1), or non-causal (0). After discussion with PNNL, we added labels for a few edge types which were not labeled in the dataset: "increases" and "directlyIncreases" (+1), "decreases" and "directlyDecreases" (-1), and "hasVariant" and "partOf" (0). Additionally, the "hasReactant" and "hasProduct" edges were reversed to "isReactantOf" (-1) and "isProductOf" (+1) to reflect their biological role in pathways. Also due to this discussion, we added an additional restriction to the templates that non-causal edges are not allowed in paths. To limit the number of results to a manageable size, we restricted the maximum path length to 3. This resulted in 68 million possible paths in total, distributed unevenly between the 1,874 templates. We plot this distribution in Figure 6. We observe that around 220 of the templates had no results, as shown in the leftmost bar of the histogram. However, it is a comparatively rare event that a template has only one possible match, with most of the templates having between 100 and 10,000 possible matches.

4.2.2C Version 3.3.1

Version 3.3.1 of the COVID dataset has the same world graph as version 3.3.0, with some minor formatting fixes. However, the template format is changed to include a potential "confounder", a new node with pathways leading from it to both the drug in question and SARS-CoV-2. These pathways are not constrained to be activating or inhibiting, but must still contain no non-causal edges. Each template comes in two variations: "support", where the pathway from the drug to SARS-CoV-2 is inhibiting as in version 3.3.0, and "refute", where the pathway is instead activating. This results in 4,420 templates in total, two each for the 2,210 possible drugs. A diagram of this template from the dataset's associated readme file is shown below.



Figure 11: Template for COVID paths for Dataset version 3.3.1.

This confounder is not specified to be a specific node or set of nodes. However, we can accommodate it by first treating it as specifying the set of all nodes. We then repeatedly apply our constraint enforcement algorithm (with edge counts) for both paths leading from the confounder. In between each step, we take the intersection of the candidates for the start points of both paths, and use this as the new set of start nodes S . We repeat this process, taking smaller and smaller sets S until no smaller set can be found. At this point, we proceed to our regular pathfinding algorithm.

Using this approach, we find a much larger set of results for the version 3.3.1 templates than those for V3.3.0 and V3.2.1. For just the first template, applying a maximum path length of 3, we identify 784,184 possible paths from the drug in question (Lopinavir) to SARS-CoV-2. Additionally, there are 2,558 possible confounders. Each confounder has multiple paths from it to the drug and to SARS-CoV-2, which leads to many possible combinations. Totalling these combinations together for all the confounders, we find that there are about 89 billion possible combinations, which when multiplied by the number of possible paths from the drug to SARS-CoV-2 yields a total of 69.8 quadrillion possible results for just the first template.

4.3 COVID Knowledge graph and structural equivalence

In [Zucker *et al* 2021], a knowledge graph is assembled from multiple sources including the COVID-19 Open Research Dataset, the Blender Knowledge Graph, and the comparative toxigenomics database. The authors of [Zucker *et al* 2021] 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, we understand the template and world to be comprised of nodes representing biochemical entities. In the template, some nodes are specified, and others are labeled as either a chemical, gene or protein. The channels (of which there are 9) in this problem are various known types of interactions between entities; for example, one chemical may activate another protein. Finding solutions amounts to determining assignments for each of

the unspecified nodes which have the desired chemical interactions. There is an abundance of solutions to this problem, and incorporating equivalence greatly enhances our ability to understand the solution space. The figure below depicts the template and Venn diagrams of candidate sets for one solution class and exposes unspecified template nodes with a large number of candidates. Such information is useful to an analyst for determining entities which may be confounding factors in a pathway as well as suggesting additional label information or interactions to add to better specify the entire solution space.

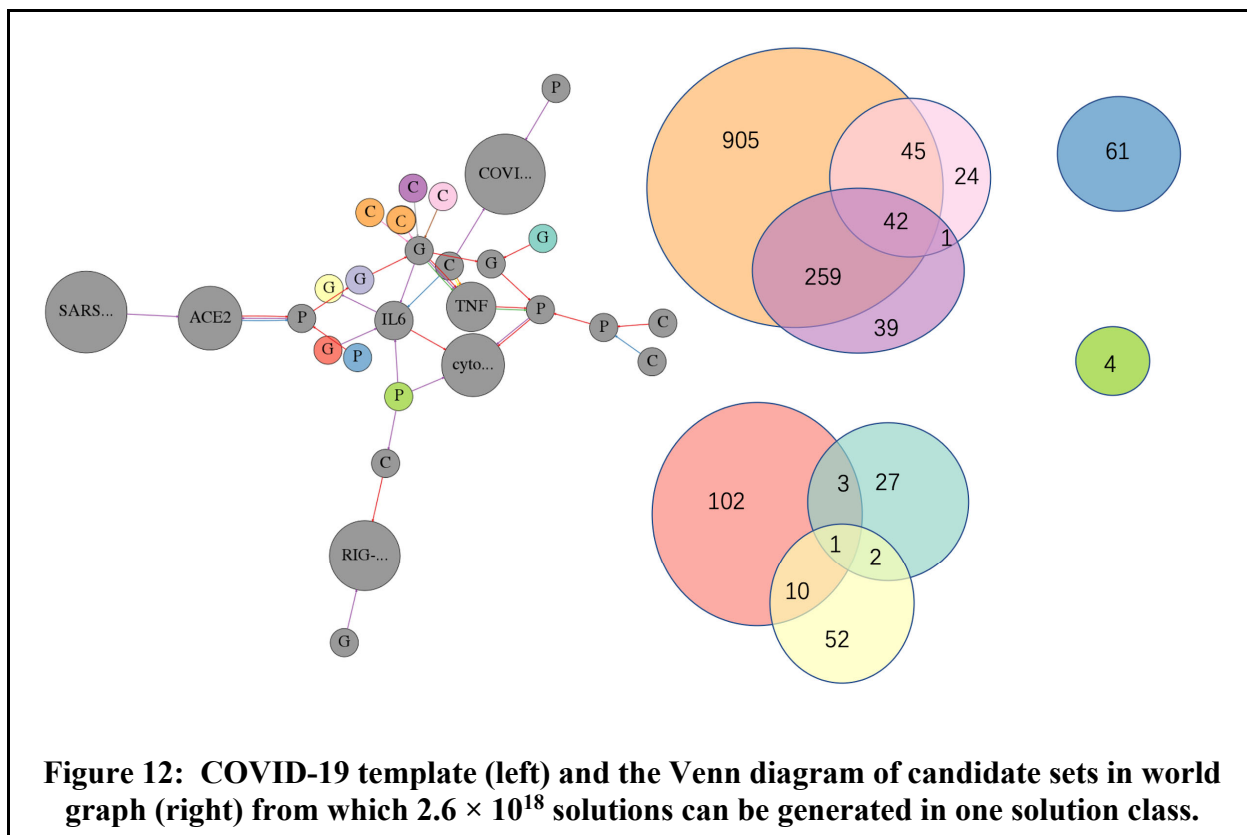


Fig 12. On the left is the COVID-19 template (left) and the Venn diagram of candidate sets in world graph (right) from which 2.6×10^{18} solutions can be generated in one solution class. Each section in the Venn diagram represents a node cover equivalence class, and the number in the section is the size 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.

5.0 CONCLUSIONS

For Phase I, we introduced a series of effective filtering methods that can be used to solve subgraph isomorphism problems within a large-scale multiplex network. Rather than solving the problem posed by DARPA for TA3, of finding a single subgraph isomorphism, we elected to try to find all isomorphisms, or as many as possible. We found that the datasets provided by the TA1 performers naturally resulted in many isomorphisms due to the combinatorial complexity of the matching problem.

The filtering approaches reduce the search space using node labels/attributes, node-level statistics, topology, and all different constraints. When applied iteratively until convergence, our filtering methods significantly narrow down the search space. We also implement a more computationally expensive isomorphism counting approach, which can be applied after filtering to solve the Subgraph Isomorphism Counting Problem. We can also proceed with a validation step that checks whether each world node participates in at least one subgraph isomorphism, hence solving the Signal Node Set Problem. We have applied our methods to multiple types of networks: Sudoku puzzles, three real-world datasets, and synthetic datasets created by PNNL, GORDIAN and IvySys as part of the DARPA MAA program. In all of these experiments, our methods have greatly narrowed down the search. When applied to the Higgs Twitter network, the commercial airline network, PNNL Version 6, PNNL Real World and GORDIAN Version 7, our methods are capable of solving most, if not all, of the proposed subgraph isomorphism problems. However, even in more difficult settings such as the Great Britain Transportation and IvySys Versions 7 and 11, our methods reduce the search space to a much smaller scale. Our results show that further narrowing down the candidates of template nodes is often hindered by an abundance of subgraph isomorphisms existing in the world graphs. Under such circumstances, leveraging more node attributes such as geospatial coordinates or timestamps can assist in making further progress.

Under this program we have also developed a theory for static and dynamic notions of equivalence and presented conditions under which vertex assignments can be safely interchanged while preserving isomorphisms. We have shown that with minimal changes to a subgraph isomorphism routine to incorporate observed equivalence during a tree search, we can dramatically reduce the amount of time to solve a problem and get a compact characterization of the solution space. For instances with minimal symmetry, little is to be gained from applying symmetry, but for problems with large symmetric structures, it is essential to exploit equivalence in order to understand the large solution space. We have demonstrated the applicability of our results to standard subgraph solvers by integrating our methods into the state-of-the-art solver Glasgow and extended our methods to the more complex problem spaces of multiplex multigraphs. Future directions for this research include adapting these notions of equivalence to inexact search as well as producing inexact forms of equivalence. We would also like to better understand how to incorporate automorphic equivalence with the different notions of equivalence discussed in this paper.

For Phase II we provide an approach for inexact subgraph matching on attributed graphs via optimization of the graph edit distance. Using an approach analogous to constraint propagation, we develop lower bounds on the cost of individual assignments, then combine them using linear sum assignment. Using these cost bounds as a heuristic, we perform a guided depth first search

for optimal solutions. We published results applying our approach to the AIDA V2.1.2 knowledge graph dataset. Our results on that dataset were state of art for the DARPA MAA program. We also developed methods for the COVID paths dataset and the COVID knowledge graph dataset. In both cases we were successful in identifying a large number of solutions. For the latter case we developed a Venn diagram and compressed graph structure for a portion of the solution space.

In both Phase I and Phase II, we discovered that subgraph matching can have a very large and complex solution space especially when the world graph has many more nodes and edges than the template. This issue is innate to the problem even with datasets are delineated by attributes, as in the case of knowledge graphs. In a real use-case scenario, analysts may need to query additional information about template nodes or world nodes to reduce the problem size and the solution space. Currently, this query process is done by hand, based on the personal experience of analysts. By analogy to the well-known active learning problem in machine learning classification problems, we believe that a machine-based active learning problem for the subgraph match problem in which the machine suggests optimal template target nodes that would be most likely to reduce the solution space when it is otherwise overly large and complex. The humans in the loop can then include additional information about those target nodes. At this end of this project we published a conference paper in which we presented some case studies for both synthetic and real world datasets for multichannel subgraph matching. We believe this should be an important area for future research.

6.0 REFERENCES

- [Bi et al., 2016] Bi, F., Chang, L., Lin, X., Qin, L., and Zhang, W. (2016). Efficient subgraph matching by postponing cartesian products. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1199–1214. ACM.
- [Bonnici et al., 2013] Bonnici, V., Giugno, R., Pulvirenti, A., Shasha, D., and Ferro, A. (2013). A subgraph isomorphism algorithm and its application to biochemical data. *BMC bioinformatics*, 14(7):S13.
- [Cardillo et al., 2013] Cardillo, A., Gomez-Gardees, J., Zanin, M., Romance, M., Papo, D., del Pozo, F., and Boccaletti, S. (2013). Emergence of network features from multiplexity. *Scientific Reports*, 3:1344.
- [Carletti et al., 2017] Carletti, V., Foggia, P., Saggese, A., and Vento, M. (2017). Introducing VF3: A new algorithm for subgraph isomorphism. *Graph-Based Representations in Pattern Recognition*, pages 128–139.
- [Carletti et al., 2015] Carletti, V., Foggia, P., and Vento, M. (2015). Vf2 plus: An improved version of vf2 for biological graphs. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 168–177. Springer.
- [Conte et al., 2003] Conte, D., Foggia, P., Sansone, C., and Vento, M. (2003). Graph matching applications in pattern recognition and image processing. In *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, volume 2, pages II–21.
- [Cordella et al., 2004] Cordella, L. P., Foggia, P., Sansone, C., and Vento, M. (2004). A (sub) graph isomorphism algorithm for matching large graphs. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(10):1367–1372.
- [Han et al., 2013] Han, W.-S., Lee, J., and Lee, J.-H. (2013). 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.
- [He and Singh, 2008] He, H. and Singh, A. K. (2008). 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.
- [Ingalalli et al., 2016] Ingalalli, V., Ienco, D., and Poncelet, P. (2016). Sumgra: Querying multi-graphs via efficient indexing. In *International Conference on Database and Expert Systems Applications*, pages 387–401. Springer.

- [Jüttner and Madarasi, 2018] Jüttner, A. and Madarasi, P. (2018). Vf2++an improved subgraph isomorphism algorithm. *Discrete Applied Mathematics*, 242:69–81.
- [Larrosa and Valiente, 2002] Larrosa, J. and Valiente, G. (2002). Constraint satisfaction algorithms for graph pattern matching. *Mathematical Structures in Computer Science*, 12(4):403422.
- [McCreesh and Prosser, 2015] McCreesh, C. and Prosser, P. (2015). A parallel, backjumping sub- graph isomorphism algorithm using supplemental graphs. In Pesant, G., editor, *Int. Conf. on Principles and Practice of Constraint Programming*, pages 295–312. Springer Int. Publishing.
- [Nabti and Seba, 2017] Nabti, C and Seba, H. (2017), Compact neighborhood index for subgraph queries in massive graphs. *arXiv preprint arXiv 1703.05547*.
- [Shang et al., 2008] Shang, H., Zhang, Y., Lin, X., and Yu, J. X. (2008). Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. *Proceedings of the VLDB Endowment*, 1(1):364–375.
- [Solnon, 2010] Solnon, C. (2010). AllDifferent-based Filtering for Subgraph Isomorphism. *Artificial Intelligence*, 174:850–864.
- [Ullmann, 1976] Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42.
- [Ullmann, 2010] Ullmann, J. R. (2010). Bit-vector algorithms for binary constraint satisfaction and subgraph isomorphism. *Journal of Experimental Algorithmics (JEA)*, 15:1–6.
- [Zampelli et al., 2010] Zampelli, S., Deville, Y., and Solnon, C. (2010). Solving subgraph isomorphism problems with constraint programming. *Constraints*, 15:327–353
- [Zhang et al., 2009] Zhang, S., Li, S., and Yang, J. (2009). 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.
- [Zhao and Han, 2010] Zhao, P. and Han, J. (2010). On graph query optimization in large networks. *Proceedings of the VLDB Endowment*, 3(1-2):340–351.
- [Zucker *et al* 2021] J. Zucker, K. Paneri, S. Mohammad-Taheri, S. Bhargava, P. Kolambkar, C. Bakker, J. Teuton, C. Hoyt, K. Oxford, R. Ness, and O. Vitek, “Leveraging structured biological knowledge for counterfactual inference: A case study of viral pathogenesis,” *IEEE Transactions on Big Data*, vol. 7, no. 01, pp. 25–37, Jan. 2021.

7.0 APPENDIX A – PUBLICATIONS AND PRESENTATIONS

A.1 Presentations

All talks are by the PI unless otherwise indicated.

Invited talk – International Congress of Mathematicians in Rio de Janeiro.

Invited talk, Mathematics of Games in the Applied Sciences, Satellite Workshop of the ICM, Aug 10, 2018, IMPA, Niteroi, Brazil

Invited presentation, Math Frontiers Webinar on 'Algorithms for Threat Detection' hosted by the National Academy of Sciences, August 14, 2018

Keynote Talk, GTA3 2.0: The 2nd Workshop on Graph Techniques for Adversarial Activity Analytics, in association with the 2018 IEEE BIG DATA conference, Seattle WA Dec 10, 2018

Workshop paper talk (Thomas Tu) on our filtering code for the DARPA MAA program, at the GTA3 workshop at the 2018 IEEE BIG DATA conference Seattle WA, Dec 10, 2018 (student paper prize).

Nov. 2, 2018, Invited Talk in Workshop on the Mathematics of Computation, ICERM Brown Univ. Providence, RI.

Oct 25, 2018 Seminar, HRL Labs Malibu

Seminar, Department of Statistics, UCLA, Oct 2, 2018

The Courant Lectures, NYU, October 15 and 17, 2018

Invited talk, ICERM Workshop on Image Description for Consumer and Overhead Imagery, February 25, 2019

Invited talk; A Conference to Celebrate the 70th Birthday of Andrew Majda: Nonlinear Dynamics and Data: Prediction, State Estimation, and Uncertainty Quantification in Complex Systems, NYU, March 15, 2019

Plenary Talk, SIAM Central States Meeting, October 19, 2019

Colloquium, Macquarie Univ., Sydney Australia, November 7, 2019

Probability Seminar, Boston University, February 28, 2019

Applied Mathematics Seminar, University of Sydney, November 13, 2019

Invited 40 minute talk, AMS-AWM Special Session on Mathematical and Computational Research in Data Science, I, Joint Math Meetings, Denver, January 17, 2020

Mathematics Colloquium, San Diego State University, February 24, 2020

Shay Deutsch paper presentation at AAAI New York Feb. 2020

One World Mathematics of Information Data and Signals (MINDS) Seminar, April 23, 2020, on zoom with over 340 attendees [youtube link](#)

Princeton Computational and Applied Math Colloquium, via Zoom, May 4, 2020

SIAM Conference on Mathematics of Data Science, Plenary Talk, June 22, 2020

SIAM Workshop on Network Science, Keynote talk July 10, 2020

Invited talk, IPAM workshop on Social Dynamics beyond Vehicle Autonomy, Dec. 1, 2020

Invited talk, Analysis on Metric Spaces workshop at Okinawa Institute of Science and Technology, December 8, 2020, on zoom

Invited talk Workshop on Mathematical Machine Learning and Application, Penn State Univ. Dec. 16, 2020, on zoom

Data Driven Methods for Science and Engineering Seminar, University of Washington Seattle, January 8, 2021

Invited talk - Faraway Fourier Talks, University of Maryland, January 25, 2021

Seminar, One World Seminar Series on the Mathematics of Machine Learning, Feb. 3, 2021

Distinguished Lecture, April 8, 2021, Univ. Waterloo, Canada

Invited talk, workshop on Analytic and Geometric Approaches to Machine Learning, Univ. of Bath UK (online) July 28, 2021

Invited talk Lorentz Center, Workshop on Working Groups for Women in Operator Theory, Leiden, Netherlands, July 29, 2021

Indam Day Rome Italy (on zoom), Sept. 3, 2021, Plenary talk, [YouTube Video](#)

Invited talk, One World Mathematics of INformation, Data, and Signals (1W-MINDS) Seminar, August 12, 2021 (on zoom)

Plenary Talk, November 23, 2021, Isaac Newton Institute, Cambridge, UK, Workshop on The Power of Women in Deep Learning (on zoom)

7.1 IEEE BIG DATA Workshops on Graph Techniques for Adversarial Activity Analytics (GTA3)

2018 - one paper presentation

2019 - two paper presentations

2020 - two paper presentations

2021 - three paper presentations

A.2 PUBLICATIONS, PREPRINTS, and PUBLISHED CODE

Dominic Yang, Yurun Ge, Thien Nguyen, Denali Molitor, Jacob Moorman, and Andrea Bertozzi, "Structural Equivalence in Subgraph Matching", 2022, submitted for publication.

Yurun Ge and Andrea L. Bertozzi, Active Learning for the Subgraph Matching Problem, GTA3 workshop, IEEE BIG DATA conference, 2021.

Thomas K. Tu, A Constraint Propagation Approach for Identifying Biological Pathways in COVID-19 Knowledge Graphs, 2021 IEEE International Conference on Big Data (Big Data), 2021, DOI: [10.1109/BigData52589.2021.9671290](https://doi.org/10.1109/BigData52589.2021.9671290).

D. Flocco, B. Palmer-Toy, R. Wang, H. Zhu, R. Sonthalia, J. Lin, A. L. Bertozzi, and P. J. Brantingham, An Analysis of COVID-19 Knowledge Graph Construction and Applications, GTA3 workshop, IEEE BIG DATA conference, 2021.

Thomas K. Tu, [*Subgraph Matching on Attributed Multiplex Networks with Applications to Knowledge Graphs*](#), PhD Thesis, University of California, Los Angeles, 2021.

Jacob Moorman, [*Stochastic Optimization and Subgraph Search*](#), PhD Thesis, 2021, University of California, Los Angeles.

Jing Qin, Harlin Lee, Jocelyn T. Chi, Lucas Drumetz, Jocelyn Chanussot, Yifei Lou, and Andrea L. Bertozzi, [Blind Hyperspectral Unmixing Based on Graph Total Variation Regularization](#), IEEE Trans. Geosci. Remote Sensing, 59(4), pp. 3338-3351, 2021.

Moorman, Jacob; Tu, Thomas; Chen, Qinyi; He, Xie; Bertozzi, Andrea, [Subgraph Matching on Multiplex Networks](#), IEEE Trans. Network Sci. and Engineering, 2021.

Baichuan Yuan, Frederic P. Schoenberg, Andrea L. Bertozzi, [Fast Estimation of Multivariate Spatiotemporal Hawkes Processes and Network Reconstruction](#), Annals of the Institute of Statistical Mathematics, 2021.

Bao Wang, Alex Lin, Penghang Yin, Wei Zhu, Andrea L. Bertozzi and Stanley Osher, [Adversarial defense via the data-dependent activation, total variation minimization, and adversarial training](#), Inverse Problems and Imaging, 15(1), pp. 129-145, 2021.

Thomas K. Tu, Jacob D. Moorman, Dominic Yang, Qinyi Chen, and Andrea L. Bertozzi, [Inexact Attributed Subgraph Matching](#), Graph Techniques for Adversarial Activity Analytics (GTA3 4.0) workshop, Proc. 2020 IEEE International Conference on BIG DATA, pp. 2575-2582.

Thomas K. Tu, Dominic Yang, "Fault-tolerant Subgraph Matching on Aligned Networks." Proc. GTA3 4.0: The 4th Workshop on Graph Techniques for Adversarial Activity Analytics, IEEE International Conference on Big Data, Atlanta, GA, Dec. 2020.

Kevin Miller, Hao Li, and Andrea L. Bertozzi, [Efficient Graph-Based Active Learning with Probit Likelihood via Gaussian Approximations](#), Workshop on Real World Experiment Design

and Active Learning, at the International Conference on Machine Learning (ICML), July 18, 2020.

Shay Deutsch, Andrea Bertozzi, and Stefano Soatto, [Zero Shot Learning with the Isoperimetric Loss](#), Proc. Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20), 34(7), pp. 10704-10712, 2020.

Zach Boyd, Mason A. Porter, and Andrea L. Bertozzi, [Stochastic Block Models are a Discrete Surface Tension](#), J. Non. Sci., 30, pp. 2429-2462, 2020; published online April 22, 2019. [ArXiv preprint](#)

Hui Jin, Xie He, Yanghui Wang, Hao Li, and Andrea Bertozzi, [Noisy Subgraph Isomorphisms on Multiplex Networks](#), Third workshop on Graph Techniques for Adversarial Activity Analytics (GTA3 3.0), IEEE BIG DATA conference, Los Angeles, Dec 9, 2019.

Thien Nguyen, Dominic Yang, Yurun Ge, Hao Li, and Andrea Bertozzi, [Applications of Structural Equivalence to Subgraph Isomorphism on Multichannel Multigraphs](#), Third workshop on Graph Techniques for Adversarial Activity Analytics (GTA3 3.0), IEEE BIG DATA conference, Los Angeles, Dec 9, 2019. **OUTSTANDING PAPER PRIZE**

Jing Qin, Harlin Lee, Jocelyn T. Chi, Yifei Lou, Jocelyn Chanussot, and Andrea L. Bertozzi, [Fast blind hyperspectral unmixing based on graph Laplacian](#), Proc. 10th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), Amsterdam, Sept. 24-26th, 2019.

Jacob D. Moorman, Qinyi Chen, Thomas K. Tu, Zachary M. Boyd, Andrea L. Bertozzi, [Filtering Methods for Subgraph Matching on Multiplex Networks](#), Proc. GTA3 workshop, IEEE Conference on BIG DATA, Seattle Dec. 2018, pp. 3979-3984.

We have published our code for the subgraph matching problem as an open-source Python package available on GitHub (<https://github.com/jdmoorman/uclasm/tree/master>) .

We have published an interactive template map for the British Transportation Network. X. He. Interactive template map. <https://hexie1995.github.io/transportation-on-the-map/global>, 2019.

We have published a multichannel adaption of the Glasgow subgraph solver that also includes structural equivalence algorithms. <https://github.com/domyang/glasgowsubgraph-solver>.

A.3 Honors and Awards

Jacob Moorman UCLA Dissertation Year Fellowship 2020-2021

Andrea Bertozzi - elected to US National Academy of Sciences 2018

Andrea Bertozzi - SIAM Kleinman Prize 2019

Outstanding Paper Prize GTA3 Workshop in IEEE BIG DATA 2019

8.0 LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

DOD	Department of Defense
DARPA	Defense Advanced Research Projects Agency
MAA	Modeling Adversarial Activity
NE	NE = No equivalence;
TE	TE= template equivalence;
WE	WE = world equivalence;
FE	FE = full equivalence;
TEWE	TEWE= template and world structural equivalence;
CE	CE= candidate equivalence;
NC	NC = node cover equivalence.
COVID-19	COVID-19 Coronavirus Disease 2019
SARSCOV-2	SARSCOV-2 Severe Acute Respiratory Syndrome Coronavirus 2
PNNL	Pacific Northwest National Laboratory
GTCB	Ground Truth Cost Bound