VAST 2020 Contest Challenge: GraphMatchMaker: Visual Analytics for Graph Comparison and Matching

Natkamon Tovanich
IRT SystemX and Université Paris-Saclay, CNRS, Inria, LISN, France

Alexis Pister, Gaëlle Richer
Université Paris-Saclay, CNRS, Inria, LISN, France

Paola Valdivia, Christophe Prieur
I3, CNRS, Telecom Paris, Institut Polytechnique de Paris, France

Jean-Daniel Fekete, Petra Isenberg
Université Paris-Saclay, CNRS, Inria, LISN, France

***Abstract*—We report on the process and design of our visual analytics graph analysis challenge winning entry. Specifically, our team addressed the IEEE VAST 2020 Mini-Challenge 1 that asked participants to identify a group of people that accidentally caused an internet outage. To identify this group, we were given a network profile and a large multi-variate social network to search in. Our approach involved statistical and graphical analysis as well as the design of three custom visual analytics tools. The submitted solution and visualizations are available at https://graphletmatchmaker.github.io/.**

■ INTRODUCTION

Mining graph (network) data is a major topic in data science and has been so for the last two decades [1]. Many methods have been devised to search large graphs for specific patterns in various applications ranging from biology [2], [3] to online fraud detection [4]. Yet, effectively using such methods in practical applications poses another set of challenges that we tackled throughout our participation in a visual analytics graph analysis challenge.

From May–August 2020, our team worked on a data analysis challenge organized annually by the IEEE Visual Analytics Science and Technology (VAST) conference. A team of VAST challenge chairs and colleagues design this challenge each year to offer researchers the opportunity to test their tools and skills on realistic tasks and datasets. We chose to participate in the first of three offered mini-challenges. Mini-Challenge 1 asked for skills in the area of graph analysis, which matched the background and experience of our team members. Specifically, our team consisted of two Ph.D. students, two postdocs, and three senior researchers in graph analysis and visualization. Our approaches were influenced by the research background of these various team members as well as the research some of us were currently working on. This team composition resulted in a diverse and ultimately quite successful approach to tackling the challenge.

The VAST Challenge 2020 [5] presents a scenario in which a group of “white hat” hackers accidentally caused an Internet outage while trying to protect the global internet from malicious cyber-attacks. The overall goal of Mini-Challenge 1 was to identify who these hackers possibly were based on an anonymous profile derived by sociopsychologists to most likely resemble the structure of the group. In Mini-Challenge 1, participants were given a large multivariate social network as their main data structure: the *large graph* as well as a smaller *template graph* representing the sociopsychologists’ anonymous profile. The network is a directed temporal graph consisting of multiple types of nodes and edges. The majority of the data represents the relationship between person nodes linked through phone call and e-mail edges and is complemented by information on demographics, publication records, procurement, and travels. Participants were also presented with possible *candidate graphs* extracted from the large graph that could represent the members of the white hat hacker group.

Our main approach is based on finding nodes that have similar node properties and edge consistency between the graphs. We present our approach and three visualization tools we built to address this challenge under the name *GraphMatchMaker*. Specifically, we present visualizations to compare the template graph and describe our methods to find matching subgraphs in the large graph. Based on the reflection of our process, we propose the MatchMaker tool to find matching node pairs between two graphs and incrementally construct the matching subgraph. Thus, the tool can be used to perform local graph comparison and matching tasks in multilayer and dynamic networks.

THE CONTEST PROBLEM

The 2020 VAST Challenge posed a series of analytic questions united under an overarching cyber threat scenario:

“*In response to an increase in malicious cyber-attacks, numerous “white hat” hacker organizations have taken it upon themselves to fight back to protect the global internet. As the attacks became more and more aggressive, one white hat group, who has so far stayed anonymous, accidentally launched a cyber event that took down the global internet. The world’s experts are having difficulty understanding what happened and need to get in touch with the group so the effects on the internet can be neutralized and services restored.*”

This year’s mini-challenges addressed three distinct problems at the intersection of visual analytics and other specialties, requiring substantial creativity not only in the development of the interactive visual analysis, but in the data transformation and machine learning approaches as well. The first mini-challenge, designed as a follow-up to a large graph challenge first presented in 2018, involved both large graph analysis and subgraph matching. The second mini-challenge asked participants to combine visual analytics techniques with image labeling and interactive machine learning, demonstrating how visual analytics can be leveraged to develop and refine machine learning models. The third mini-challenge focused on design rather than implementation, asking participating teams to envision an interface to support both real-time situation awareness and response team composition.

VAST Challenge 2020 Co-chairs:

Kris Cook is a technical advisor at Pacific Northwest National Laboratory. Contact her at kris.cook@pnnl.gov.

R. Jordan Crouser is an assistant professor of Computer Science at Smith College. Contact him at jcrouser@smith.edu.

CONTEST EVALUATION

The 2020 VAST Challenge received 24 completed submissions from teams at 22 institutions in 6 different countries. More than 40 individual reviewers from both the visual analytics research community and subject matter experts volunteered their time and expertise, evaluating submissions on their application of visual analytics, as well as their overarching analytic approach. Each submission received detailed reviews from at least three external reviewers, and each was also reviewed by the members of the VAST Challenge Committee. Each reviewer received a written briefing providing relevant context and background information for this year's challenge scenarios, descriptions of known anomalies and patterns embedded within the data, and sample solutions to each mini-challenge. Individual reviewers provided both a numeric score and a detailed narrative describing each submission's strengths, as well as opportunities for improvement or further refinement. Additionally, each reviewer was invited to nominate exemplary submissions to the committee for award consideration. Though submissions that correctly identified known signals within the data often received favorable ratings, other factors such as creativity, clarity, and novelty were also considered in determining a submission's final score.

Of the 24 completed submissions, 3 were selected to receive awards, and an additional 5 received honorable mentions. The 2020 committee paper [5] details the award-winning submissions, as well as provides a more in-depth discussion of the challenge itself. The committee selected *"GraphletMatchMaker: Visual Analytics Approaches to Graph Matching in Cybersecurity Communities"* as one of the contest's most exemplary submissions, earning an award for their Outstanding Comprehensive Mini-Challenge 1 Solution. The VAST Challenge has a long history of providing opportunities for cross-institutional collaboration in the study of visual analytics. The attached article is an extended discussion of this interdisciplinary team's results.

Table 1. List of node and edge types for each channel in the white hat hackers network.

Channel	From Node	Edge	To Node
Communication	Person	Call to	Person
	Person	Mail to	Person
Procurement	Person	Sell	Product
	Person	Buy	Product
Co-authorship	Person	Write	Document
Travel	Person	Travel to	Country

Details about the Data

Mini-Challenge 1 contained data for one year of activities of pseudonymous white hat hackers as a large graph of 7.4GB of data stored in 123,892,863 records. The graph contains five node types and seven edge types associated with four channels as outlined in Table 1. The dataset, therefore, has been modeled as a multivariate multilayer dynamic network with bipartite layers, making it heterogeneous.

During the data exploration phase, we made several observations about the graph based on additional information:

Financial profiles: Each person had a financial profile that consisted of his/her total financial income or spending in 29 categories (e. g., electricity, healthcare, transportation, groceries, personal taxes).

Location: No information on the home location for each hacker was given. Yet, each person node had a single location for their outgoing phone calls (phone edges). Therefore, we took this location attribute for each person as their origin "location".

Procurement: Most buy edges had a relative sell edge connected to the same product and with the same timestamp.

Authorship: People who wrote the same document are considered co-authors. This kind of relationship was rare in the data.

The data also consisted of an anonymous profile, the *template graph*, that was presented as most likely resembling the structure of the white hat hacker group that caused the internet outage. It consisted of 88 nodes and 1,325 edges with the same data format as Table 1. Node IDs were anonymized and did not match directly with the large graph. Moreover, challenge participants were given five *candidate graphs*, subgraphs extracted from the large graph with approximately the same number of nodes and edges. These candidates were starting points that had to be compared to the template before moving on to the large graph.

DEFINING ANALYTICS TASKS

Based on the scenario description and our first exploration of the data structures, we abstracted the challenge into two tasks we needed to tackle:
Task 1: Small graphs comparison: Compare the template graph with each candidate graph and select the most similar candidate to the template. Explain where the two graphs agree or disagree.
Task 2: Subgraph matching: In the large graph, find the subgraph that most resembles the template. Start from the provided *seeding edges* in the large graph.

Our work on these two tasks was non-linear. Team members explored different approaches in parallel, helping each other with questions and information. The following sections provide details on what we tried before discussing the success or failure of our attempts to solve the tasks.

SMALL GRAPH COMPARISON

To find out which candidate graph best compared to the template, we started by both visualizing the graphs and analyzing them using graphlets. To visualize the data, we designed multiple visualizations that highlight different perspectives of the data. We drew node-link diagrams to compare the overall graph structures, graphlet frequencies to understand node connectivity patterns, and visualized temporal profiles to observe hacker activities in each graph over time.

Node-link view

To first compare graph structures at a high level, we visualized the template and the five candidate graphs as node-link diagrams using a simple force-directed layout to assess their structural similarity visually. The visualization combines the graphs in each channel into a single graph and uses node color and shape to encode node location and type, respectively (Figure 1). We considered location an important hacker attribute and assigned it a strong visual encoding (hue). We also added checkboxes to filter edge types and explore whether certain edges led to clearer similarities than others.

We first inspected the template graph in detail. Three specific communities emerged: the first was a dense community of hackers connected through the communication channel (red polygon in the figure). The second community was smaller and connected to the first group through a bridge person (blue polygon). The third group was formed by people connected by traveling to the same location. Next, we looked at the candidate graphs: Candidates 1 and 2 showed a structure and communities similar to the template, while Candidate 3 had more sparse communication edges. The other candidate graphs (4 and 5) did not seem similar visually. From this view, we narrowed our answer down to Candidate Graphs 1 and 2. Next, to compare the graphs, we analyzed their topological structures using graphlets.

Graphlet view

We conducted a graphlet analysis (see the *Graphlet* sidebox) to measure the topological similarity between the template and candidate graphs via the communication channels (phone and email). We assumed these channels to be mutual between the connected hackers, so we decided to use undirected graphlets. We chose to study 5-node graphlets as 4-node graphlets cannot capture complex patterns, and 6-node graphlets are too numerous to compare.

The graphlet frequencies of each graph were computed using an optimized version of the gTrieScanner library [9] to count occurrences of graphlet patterns in a given graph. Graphlet frequencies indicate the connectivity patterns in a graph: higher normalized frequencies mean that the nodes in the graph are more connected in a particular pattern. Figure 2 (A) shows histograms that allowed us to compare the normalized graphlet frequencies of the template and all candidates. From the figure, the graphlet frequencies are highly different for each graph—showing distinct signals of connectivity patterns.

Many similarity measures exist to compare vectors of frequencies. In this work, we chose to use *Pearson's correlation coefficient* because it is scale independent and is one of the easiest measures to interpret as the values are in the limited range of [-1, 1].

Using this measure, Candidate 2 was the most similar to the template with a correlation of 0.761,

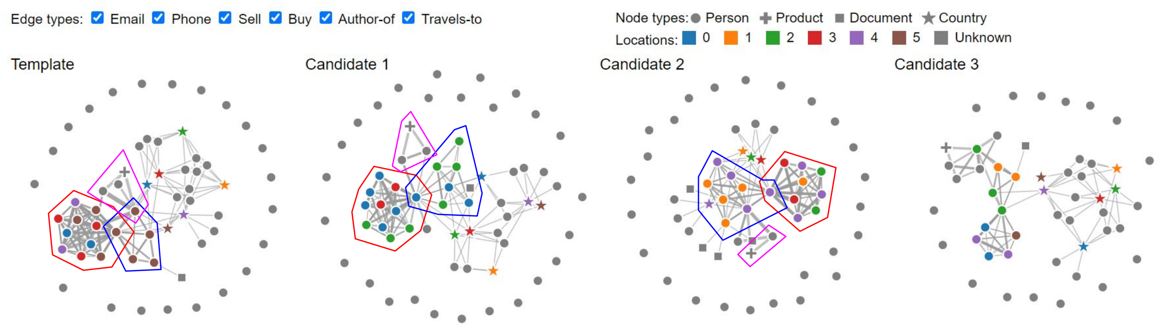

Edge types: ✓ Email ✓ Phone ✓ Sell ✓ Buy ✓ Author-of ✓ Travels-to
Node types: ● Person ✚ Product ■ Document ★ Country
Locations: 0 1 2 3 4 5 Unknown
Template
Candidate 1
Candidate 2
Candidate 3

Figure 1. Node-link diagram view. Node shape corresponds to node type, while color indicates node's location. Edge types can be filtered with checkboxes. Candidate graphs 4 and 5 can be found at https://graphletmatchmaker.github.io/AVIZ-Tovanich-MC1/node-link-view/networks.html.

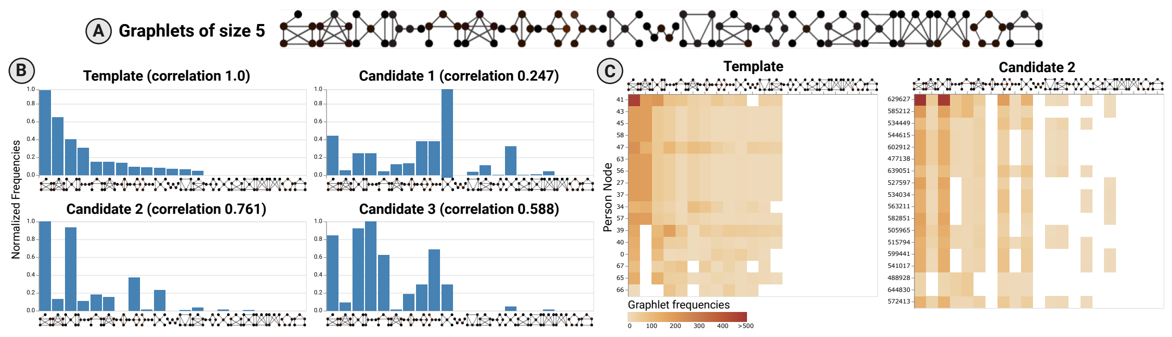


Figure 2. Graphlet frequency view. (A) Graphlets of size 5 sorted by frequencies in the template graph. We used this order on the x-axis of both charts. (B) Relative frequency of each graphlet as a histogram. (C) Graphlet frequencies for each node as a heatmap. Nodes in the y-axis are sorted by eigenvector centrality.

followed by Candidate 3 with a correlation of 0.588. The template and Candidate 2 shared the same most frequent graphlet pattern corresponding to the clique-minus-one . While Candidate 3 also had a high frequency for this pattern, it was not the most frequent. The second most frequent pattern of the template was the clique , but none of the candidates showed similar values. The result demonstrated that even if Candidates 2 and 3 had a high correlation, there were still significant discrepancies between the structures of the template and all the candidates according to the graphlet frequencies.

We also looked at the graphlet frequencies for each node in the graph as a signature describing each hacker's communication pattern. Two nodes with similar graphlet frequencies suggest that they might correspond to the same person. Figure 2 (B) shows two heatmaps to compare the graphlet frequencies of each node between the template (left) and Candidate 2 side-by-side. We see that the most common graphlet patterns among the template nodes were the clique , indicating dense communication within the group, and the clique-minus-three , connecting the dense group with an outside node. On the other hand, Candidate 2's nodes tended to have more clique-minus-three and bow-tie patterns than the template, indicating a bridge connecting two groups that we can also observe in Figure 1.

Temporal view

Time was the last graph characteristic we investigated, focusing on the intensity of connectivity in the graphs over time. Because the graph data consisted of multiple edge types, we visualized the temporal information at different

GRAPHLETS

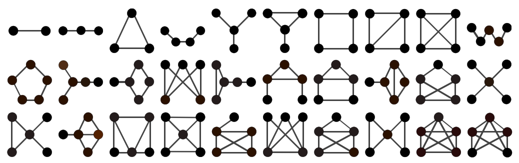


Figure 3. List of the 30 undirected graphlets from size 2 to 5.

Graphlets are small connected *induced*, *non-isomorphic* subgraphs composing any network. In an *induced* subgraph, two vertices linked in the original graph remain linked in the subgraph. For instance, if the original graph is a triangle we can only induce the simple edge or triangle subgraph (graphlet). The path of length 2 has all vertices of the original graph but misses an edge and is, therefore, not a possible graphlet. They were first introduced by Milo et al. [2] to explore the structural differences between biological networks, but they are now used in several disciplines involving networks. Usually, researchers use graphlets of a predefined size, like 3, 4, or 5, since the number of possible graphlets grows exponentially with the graph size, making the interpretation of the results and the computation harder. Graphlets can be undirected or directed–the same way networks can be. In a typical graphlet analysis, the frequency of each graphlet is computed and normalized, providing a distribution of the graphlets which occur in a graph.

Graphlet frequencies can be computed on an entire graph or around a specific node (*ego networks*) to compare the frequencies of each node [6]. This is mostly done by computing similarity measures that grasp the differences between the distributions of the graphlet frequencies [7].

Finally, a fair amount of research has been done to efficiently compute graphlet frequencies [8] since it is a problem with an intrinsically hard computational complexity.

aggregation levels. We started with an overview of the graphs to visually compare the graphs and decide what to analyze next in more detail. Then, we created detailed visualizations to focus on potential patterns of interest.

First, we created an aggregated time series view to quickly compare the template graph and the candidate graphs. We counted the number of edges per day for each graph. To quantitatively compare the temporal pattern among graphs, we computed *dynamic time warping* [10] between every graph pair and conducted a hierarchical clustering to group similar time series using the *dtaidistance* library in Python [11]. Dynamic time warping is a standard measure to find the optimal distance between two time series, which may have time shifts in the patterns [10]. Figure 4 (A) shows the time series of each graph ordered by hierarchical clustering similarity. According to the figure, Candidate 2 was the closest to the template.

Then, we decided to split this information for each type of edge and created a stacked area chart, shown in (B). From this visualization, we confirmed again that both Candidates 1 and 2 were more similar to the template than the others because: (1) They tended to have a similar number of edges over time; (2) They had the same bursts of activity throughout the year; and (3) All of them had a peak of communication activity (phone and emails) followed by only travel activity. We also observed a 14-day time shift between the burst activities in the template compared to Candidates 1 and 2.

To decide which one of the two candidate graphs, Candidates 1 or 2, was more similar to the template, we created a visualization focused on the edges per node, as shown in (C). Each horizontal line represents a node on the timeline in the x-axis. Each edge is represented as a dot colored by type. We observed that the *buy* and *sell* edges (resp. light-green and red dots) had a similar profile for both Candidates 1 and 2. None of the graphs consistently matched the travel edges in the template. Even if not conclusive, the communication bursts from Candidates 2 were more similar to the template.

Finally, we created an egocentric visualization, including all of a node's incoming and outgoing edges, to see the temporal activity of nodes in detail. In particular, we used this visualization to

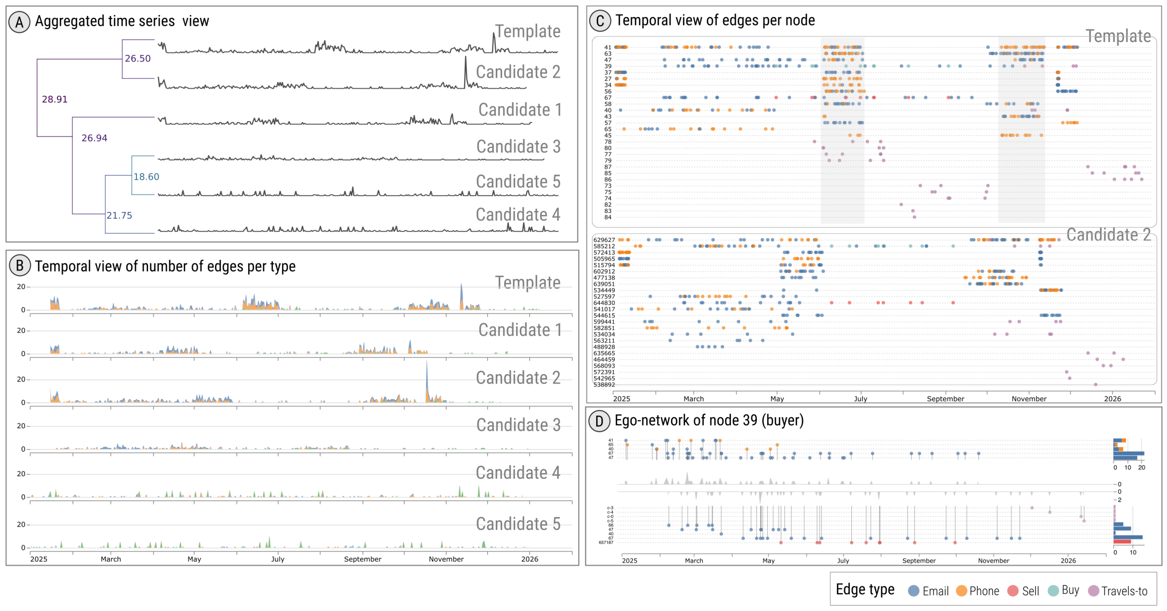


Figure 4. Temporal profile view at different levels of aggregation. (A) Aggregated time series for the template graph and candidate graphs. (B) Stacked area chart with the number of edges per type for the template graph and candidate graphs. (C) Temporal view of edges per node for the template graph and Candidate Graph 2. (D) Detail of the in (top) and out (bottom) edges for a single node of the template graph. The full resolution of these charts can be found in our report: https://graphletmatchmaker.github.io/AVIZ-Tovanich-MC1/index.htm.

see detail about specific nodes we identified as "interesting" in other visualizations. For example, Figure 1 and Figure 4 (C) showed that only one person in the template graph bought a product. We called this node the *buyer*. The visualization of its ego network is shown in Figure 4 (D). On the top (resp. the bottom) of the visualization, we drew a line for each node with an edge going into (resp. edge going out from) the *buyer* over time. Time goes from left to right along the x-axis. Each edge is represented as a dot colored by type. In the middle, between the incoming and outgoing edges, we show a time series aggregating the number of in and out edges for the *buyer*. On the right side of the figure, the bars indicate the number of edges per node. The visualization allowed us to see, for example, that the *buyer* purchased the same product several times. We also saw email exchanges with the person who sold the product.

EXTRACTING A MATCH FROM THE LARGE GRAPH

The second task in the challenge involved finding a subgraph in the large graph that matches the template graph. Our approach was based on several similarity measures to find hackers in the large graph with the same characteristics as hackers in the template graph. We developed a custom visualization: a node-matching view to help us evaluate extracted subgraphs. We also proposed two methods to extract subgraphs using the seeding edges and narrow the search space without any given edge.

Node profile similarity measures

Given that we were supposed to find a group of hackers, we focused our analysis on person nodes. We chose four characteristics of person nodes with similarity measures to compare person nodes between the template and the large graph.
Financial profile similarity: Each person node was associated with net income or spending in 29 categories. We adopted the *cosine similarity* metric to calculate the similarity between the normalized financial profiles of each person node.
Graphlet similarity: We considered the graphlet frequencies vector for each person's communication network to signal the person's ego-network

connectivity pattern. However, the algorithm to count graphlet frequencies is computationally expensive in the large graph. We developed a technique to sample the graph and estimate the graphlet frequencies vector, allowing us to compute a graphlet-based similarity measure. The sidebox gives more detail on our approach (see *Computing Graphlets in the Large Graph* sidebox).

Traveling profile similarity: The travel channel provided data about a person's trips from an origin location to a destination location, including date and duration. To construct a travel profile, we created tuples of trips (origin location, destination location) and appended them to a set of trips for each person. We applied the *Jaccard similarity coefficient* to evaluate the similarity of trips made between two people. At first, we did not consider time overlapped between two people at the same destination simultaneously. Later, we found out that this fact was very useful to narrow down the possible node matching in manual search.

Temporal activity profile similarity: We counted the number of edges associated with a person per day to build each person's temporal activity profile. We constructed two temporal profiles: (1) for the communication channel and (2) for all channels combined. We used *dynamic time warping* to calculate the similarity of the two temporal profiles between person nodes.

Node-matching view

We developed a custom visualization called node-matching view shown in Figure 6. The view consists of a heatmap matrix to display similarity measures between nodes and arc diagrams to show the connections between them, similar to the Arc Diagram-HEDA [14]. Using this view, we can evaluate the one-to-one pairing of the template's person nodes with those of the candidate graph.

The heatmap matrix (A) gives an overview of the node agreement based on similarity measures. Each node pair is represented by a column framed by the template node (top) and the matched candidate node (bottom). Each row represents a similarity measure using a blue color scale ranging from entirely similar (dark blue) to entirely dissimilar (white).

The arc diagrams (B) give an overview of the edge agreement based on the connections between

COMPUTING GRAPHLETS IN THE LARGE GRAPH

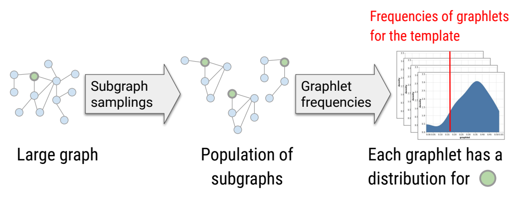

Frequencies of graphlets for the template
Subgraph samplings
Graphlet frequencies
Large graph
Population of subgraphs
Each graphlet has a distribution for ●

Figure 5. Computing pipeline for a graphlet similarity measure between a template and a large graph. The green node is a specific node of the large graph.

Computing graphlet frequencies on a large graph can rapidly become a computing resource bottleneck. To avoid this computing problem, we derived a new approach to compute a graphlet-based node similarity measure between the nodes of a relatively small graph and a large graph. For this, we approximated the relative graphlet frequencies for each node in the large graph. The pipeline is depicted in Figure 5.

Our process is based on the popular idea of extracting a population of subgraphs from the large graph [12], with sizes and structures similar to the small graph. Each node of the large graph should appear in several of these subgraphs. To be sure of that, the sampling algorithm should start from every node at least once. However, many methods exist for graph sampling. To keep the structure of the large graph in the sampled subgraphs, we used a random walk algorithm with transition probabilities weighted by the in-degrees of the nodes. It means that nodes with a high degree will be encountered more often than more isolated nodes with a low in-degree.

Once the sampling is done, it is possible to compute the graphlet frequencies of the extracted subgraphs without the computation cost exploding. For each node, we obtained frequency distributions for each graphlet pattern that appeared in the sampled subgraphs. We used the Vysochanskij–Petunin inequality [13] to test if the frequency of a graphlet in the subgraph is part of the previously computed distribution. The ratio of positive tests over negative tests (one test is done for each graphlet) is a similarity measure.

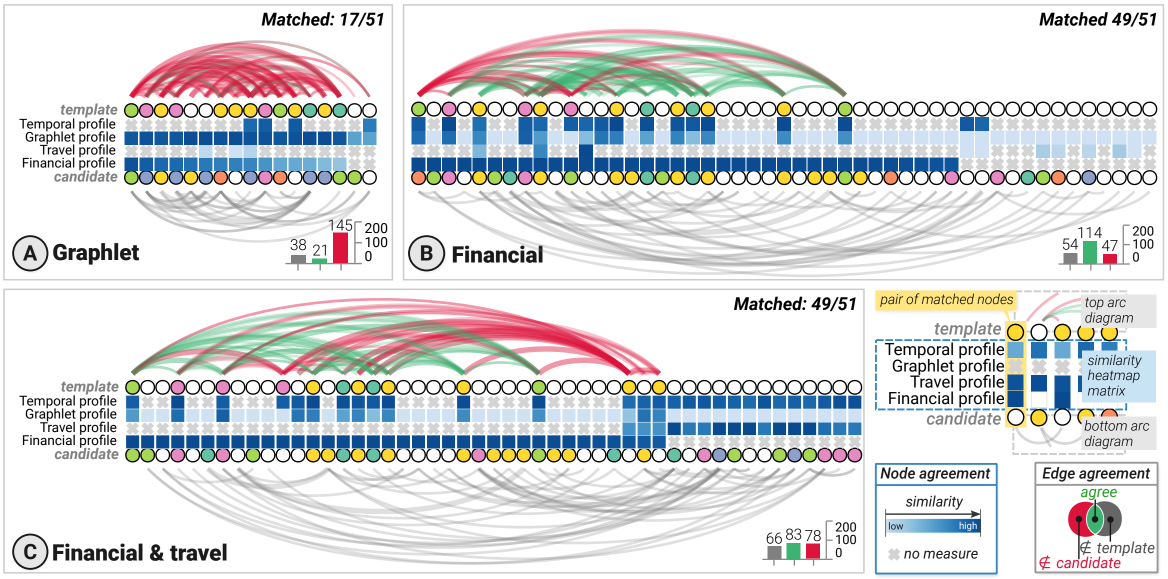


Figure 6. Node-matching views for the subgraphs resulting from the greedy matching algorithm from seed 1 using different similarity measures: (A) graphlet profile, (B) financial profile, and (C) financial and travel profiles combined. Node-matching views allow us to evaluate the agreement between the template graph and a matching subgraph. The diagram has three elements: (1) The template nodes on the top and the matching subgraph nodes on the bottom. In this challenge, the node's color indicates the person location, derived from its phone calls, and edges are from the communication channel; (2) The inner heatmap matrix shows node agreement according to the four similarity measures; and (3) Arc diagrams on the top and bottom show matching edges (green), missing edges (red), or additional (grey) edges between the two graphs. We additionally show the number of matching nodes on the top left of the chart. The bottom right holds a bar chart that shows the number of edges for each type. The demo can be found at https://graphletmatchmaker.github.io/AVIZ-Tovanich-MC1/person-pairing-view/comparison.html.

paired nodes on both graphs. The arcs on the top represent the edges of the template graph. An arc's color indicates whether the edge is matched in the candidate graph (green, true positive) or missing (red, false negative). The arcs on the bottom show the additional edges of the candidate that have no match in the template (grey, false positive). To reduce the number of edges, the edge agreement assessment does not consider the location and time of the original edge.

From this view, we can assess the quality of a candidate matching based on three criteria:

1) *Node coverage:* how many template nodes are matched (counter on the top-right),
2) *Node similarity:* the similarity of node pairs for the chosen similarity measures (we want mostly dark cells in the heatmap), and
3) *Edge matching:* how many template edges exist in the candidate graph (high number of green arcs on the top arc diagram and few grey arcs on the bottom arc diagram).

We used this node-matching view to visually assess matches for the template extracted from the large graph.

Matching based on a seeding edge

As part of the challenge, we were given three seeding edges as starting points to find suitable matching subgraphs in the large graph. With seeding edges, the matching task can be generalized to solving the maximum bipartite matching problem between nodes in the template and the large graph. We extracted the person node from the seeding edge (referred to as the seeding node) and used similarity measures to compare with person nodes in the template graph. Because there were many nodes in the large graph, it was impossible to compute all possible matchings and find the best matching subgraph. We, therefore, modified a greedy matching algorithm to dynamically add the large graph's person nodes to match the template as described in Algorithm 1. The idea is to iteratively match the nodes in the template with a set of candidate nodes of the large graph, initialized as the seeding node and its neighbors (line 3). At each step, the pair of nodes with the highest similarity value is matched together (lines 5 and 6), and the nodes are removed from the candidates (lines 7 and 8). Then the set of candidates in the large graph is extended with the neighbors of the latest matched node (line 9). This process is repeated until every person from the template matching set is coupled with a large graph person. The template matching set is only constituted of people whose similarity values are defined for the chosen similarity measures.

Since not every node in the template had information on every channel, we combined similarity measures as different similarity functions and used them to extract subgraphs with the algorithm. We tried three different functions: (1) graphlet similarity, (2) financial profile similarity, and (3) the average of financial and travel profile similarities. After the extraction, we added all other node types, such as products and locations, connected to the person nodes in the matching subgraphs.

We evaluated the subgraphs extracted from the algorithm and different similarity measures with the node-matching view. Figure 6 shows the result graphs obtained from Seed 1. Seed 2's person node did not connect to any other person by communication, so we could not use this seed as a starting point. The subgraphs extracted from Seed 3 were very similar to Seed 1.

From Figure 6, we can visually evaluate that the subgraph extracted from the graphlet similarity function (A) does not provide a good match with the template graph. The algorithm matched only 17 out of the 51 nodes, indicating a low node coverage of the subgraph. Although the financial profile similarity between the two graphs seems high, the overall edge agreement is low. Most of the retrieved edges are not matched in the template. There are a lot of missing edges (red) and a few numbers of matching edges (green) and extra edges (grey).

The subgraphs extracted using financial profiles (B) and the combination of financial and travel profiles (C) shows 49 out of 51 matching nodes, indicating a high node coverage for both subgraphs. We can see that these subgraphs have a higher number of matching edges than the graphlet subgraph. However, it is not easy to judge which one is better visually. We counted the number of edges in each color, as shown in the bottom right of the figure. The subgraph from the financial profile has a higher number of matching edges than the combination of financial and travel profiles. The numbers of missing edges and the extra edge are lower. Therefore, we conclude that the financial profile provides the best matching subgraph for the template graph out of the three similarity functions using this algorithm.

The result showed that the graphlet profile similarity did not provide a good subgraph in this challenge. Instead, using financial profile similarity led to a better solution. However, it still did not match exactly with the template. We carefully looked at the large graph and found that the person nodes were generally more densely connected than in the template. Indeed, the template was sampled from the large graph. This explains why subgraphs extracted using the algorithm had a high number of matched nodes but many additional edges (grey color on the bottom arc diagram). To reduce the false-positive error rate, we need to filter the edges that are not matched with the template graph. In the next subsection, we proposed a strategy to manually find matching nodes and filter edges in the large graph.

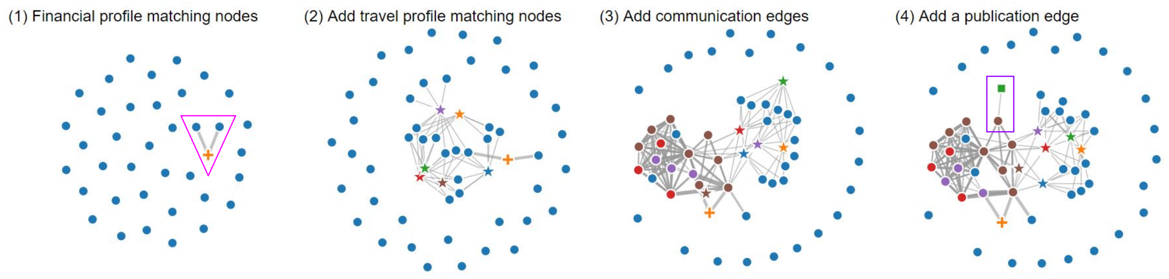
Manual node matching in the large graph

Without a seeding node, it became very challenging to find the template structure in the large graph. We cannot generate all possible subgraphs and evaluate them one by one. Consequently, we investigated the large graph and proposed a "narrow down and conquer" strategy to find the matching subgraph. We show the step of manual node matching in Figure 7.

From the exploratory analysis performed to solve Task 1, we had identified a node with a record of purchases from the same seller in the template (Figure 4 (D)). We chose this buyer pattern and looked for person nodes who purchased several products from a single person in the large

Algorithm 1: Subgraph matching with a seeding node.

```
Inputs : $G_T = (V_T, E_T)$ the template graph
         $G = (V, E)$ the large graph
         $v_0$ the seeding node.
         $SIM = [sim_{i,j}]_{i \in V_T, j \in V}$ the similarity matrix between nodes of the template and of
the large graph.
Result: matches, a list of pairs in $V_T \times V$.
1  matches $\leftarrow$ [];
2  to_match$_{G_T}$ $\leftarrow V_T$;
3  to_match$_G$ $\leftarrow set(v_0, Neighbors(v_0, G_T))$;
4  while to_match$_x$ is not empty do
5      (best$_{G_T}$, best$_G$) $\leftarrow$ idxmax($[sim_{i,j}]_{i \in \text{to\_match}_{G_T}, j \in \text{to\_match}_G}$);
6      matches.append((best$_{G_T}$, best$_G$));
7      to_match$_{G_T}$.remove(best$_{G_T}$);
8      to_match$_G$.remove(best$_G$);
9      to_match$_G$.extend($Neighbors$(best$_G$, $G$));
10 end
```



(1) Financial profile matching nodes
(2) Add travel profile matching nodes
(3) Add communication edges
(4) Add a publication edge

Figure 7. The steps of manual node matching in the large graph.

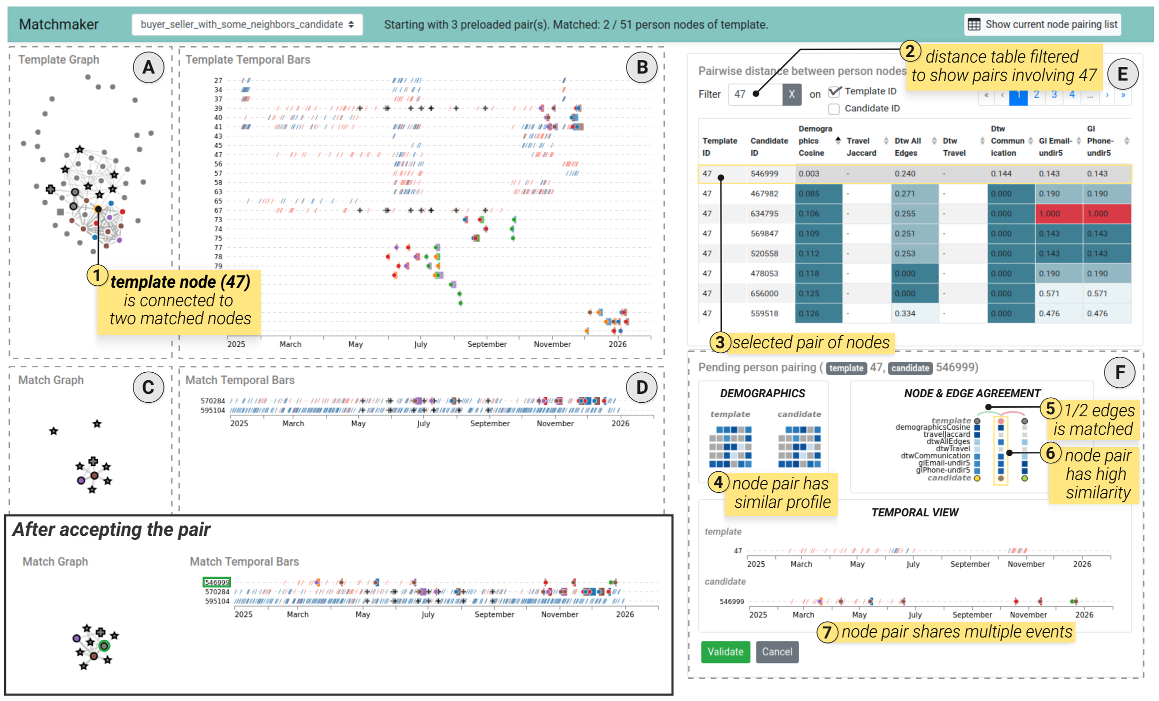
graph. We manually looked for nodes in the large graph with a similar financial profile and temporal activity and used those nodes as a starting point to extract the subgraph.

In the large graph, we found 3 buyer candidates with more than 7 purchases. In particular, we observed that one of the candidates' temporal activity was very similar to the buyer in the template graph with a 14-day time shift (Figure 7:Step 1 in the pink triangle). We also observed that the financial profiles matched well. Based on this finding, we extracted the person nodes with the most similar financial profiles and found more matching nodes with the same travel profile (Step 1).

We used node-link diagrams (Figure 1) and temporal profile charts (Figure 4) to filter edges that matched the template graph. We added the edges that shared the same travel location first (Step 2) and then added the communication edges among nodes (Step 3). Finally, we found a person with one publication and a financial profile similar to the publishing person in the template graph and included that publication edge to the matched subgraph (Step 4). This subgraph also had a very high edge agreement when visualized using the node-matching view.

MATCHMAKER TOOL

While working on the challenge, we incrementally developed a visual analytics tool to join the visualizations and measures that were useful in finding the challenge answer. The Matchmaker tool allows analysts to test different matching strategies quickly (e. g., prioritizing one node characteristic or another). Using the tool, analysts build a match for the template graph by finding



Matchmaker
buyer_seller_with_some_neighbors_candidate
Starting with 3 preloaded pair(s). Matched: 2 / 51 person nodes of template.
Show current node pairing list
Template Graph
A
1 template node (47) is connected to two matched nodes
Template Temporal Bars
B
Match Graph
C
Match Temporal Bars
D
After accepting the pair
Match Graph
Match Temporal Bars
Pairwise distance between person nodes
2 distance table filtered to show pairs involving 47
E
Filter 47 X on Template ID Candidate ID
3 selected pair of nodes
Pending person pairing (template 47, candidate 546999)
F
DEMOGRAPHICS
template
candidate
4 node pair has similar profile
NODE & EDGE AGREEMENT
5 1/2 edges is matched
6 node pair has high similarity
TEMPORAL VIEW
7 node pair shares multiple events
Validate
Cancel

Template ID	Candidate ID	Demographics Cosine	Travel Jaccard	Dtw All Edges	Dtw Travel	Dtw Communication	GI Email-undir5	GI Phone-undir5
47	546999	0.003	-	0.240	-	0.144	0.143	0.143
47	467982	0.085	-	0.271	-	0.000	0.190	0.190
47	634795	0.106	-	0.255	-	0.000	1.000	1.000
47	569847	0.109	-	0.251	-	0.000	0.143	0.143
47	520558	0.112	-	0.253	-	0.000	0.143	0.143
47	478053	0.118	-	0.000	-	0.000	0.190	0.190
47	656000	0.125	-	0.000	-	0.000	0.571	0.571
47	559518	0.126	-	0.334	-	0.000	0.476	0.476

Figure 8. Manual node-matching tool. On the left, node-link diagrams and temporal views give an overview of the template (A, B) and the current state of the match graph (C, D). On the right, analysts can sort, search and select possible node pairs from the pairwise similarity view (E). Once a pair is selected, the comparison views (F) allow the pair's node and edge agreement to be investigated relative to the current match graph, i. e. list of validated pairs.

promising node pairs and inspect them in detail (human-in-the-loop analysis). It could work on any multi-layer and dynamic networks, assuming that multiple measures of node similarity have to be considered. The detailed views we developed to compare a candidate node pair are currently specific to the challenge data (e. g., travel activity, financial profile). However, it is easy to swap them out for measures present in other datasets.

When using the tool, the analyst starts by choosing an initial set of paired nodes. In our case, these paired nodes are candidate matches for the *buyer-seller* pattern and act as different entry points the analyst can choose from to start creating a match.

The tool consists of multiple linked views, shown on Figure 8. The left panel shows an overview of the template and the subgraph being built using the node-link view and temporal view. This overview also highlights which nodes of the template graph are already matched. The right panel consists of the pairwise similarity view and the node comparison views. On the top, the pairwise similarity view is a sortable table that contains similarity measures between all pairs of person nodes. Upon selection of a node pair in the table, 3 node comparison views appear: a side-by-side view of the demographic data as heatmaps; the agreement of the pair following the encoding of the node-matching view; and a side-by-side temporal views for the pair. Together, these views allow analysts to identify node pairs with high node similarities: demographic, communication, and temporal data. With this information, the analyst validates whether the node pair is a proper match. If so, the node appears on the left panel, together with the edges associated with it. The analyst continues this process until all (or almost all) template nodes are matched.

In Figure 8, we show a step-by-step example of how one may choose which template node to focus on and how to find a proper match for it using the tool. We started with an initial node pairing of two nodes, based on a *buyer-seller* pattern (see C and D). Then, we focused on template node 47 since it is connected to both the *buyer* and the *seller* (Step 1) and consequently filtered out the table to show the corresponding candidate pairs (Step 2). We went through multiple pairs with good scores, looking for one with some edge agreement. The selected pair (Step 3) shows a very similar financial profile (Step 4), one edge matched out of the two (Step 5), and has multiple events in common (Step 6), which suggests it is a good match, especially compared to the other possible pairs inspected so far.

EVALUATION OF OUR APPROACH

After the end of the review period, we obtained the correct answer from the challenge organizers to evaluate our results.

Small graph comparison

We concluded that Candidate 2 was the best match compared to the template, followed by Candidate 1 based on three views we produced. However, the correct answer is Candidate 1, followed by Candidate 2.

A closer look at the node-link view might have helped us derive the correct answer. Inspecting the bridging nodes between the two communities carefully and counting the number of nodes and edges in each community would have helped to get the correct answer. In the graphlet and temporal views, we mainly focused on communication as the most active channel in the network data. However, the solution highlights a difference in edge counts in publication and procurement channels, which we did not consider in these views. Here, the graphlet method did not work well. Each graph had a very different frequency pattern because the graphs were intentionally manipulated by sampling and removing edges.

Subgraph matching

The organizers provided a set of node IDs that constitutes the solution subgraph. We evaluated our subgraphs using the precision, recall, and F1-measure metrics in Table 2.

- *Precision:* How many nodes we identified correctly compared to all nodes in our subgraph,
- *Recall:* How many nodes we identified correctly compared to all nodes in the solution, and
- *F1-measure:* The harmonic mean of precision and recall.

Table 2. Performance metrics of matched subgraphs using a seeding edge and manual matching.

Method	Precision	Recall	F1
Graphlets	0.604	0.330	0.426
Financial profile	0.699	0.659	0.678
Financial & travel profile	0.762	0.727	0.744
Final matching	1.000	1.000	1.000

For subgraphs extracted from the seeding nodes, the combination of financial and travel profiles provided the best result, followed by the financial profile solely. This result highlights that node attributes' similarity played a large role in finding the matched graph for this challenge.

Graphlet similarity produced the worst match. We investigated the data in detail and found that the template graph was a sampling of edges from the large graph. In this case, the graphlet approach did not work well because it assumes that the structure of nodes in the template and large graph is similar. Graphlet frequencies are sensitive to the edges in the graph [15]. By adding or removing edges in the template graph, the graphlet frequencies can change significantly and differ from the true distribution in the large graph. As a result, the graphlet structure of nodes in the large graph was largely different from the template, leading to the worst result compared to other similarity measures.

The nodes retrieved using manual matching matched precisely with the solution as our extracted subgraph matched the template graph almost perfectly. According to the result, knowing the underlying assumption of the graphs is the first priority. We tried out different strategies without knowing the assumption and eventually found that node similarities led to the correct solution. Based on this need, we developed the MatchMaker tool to discover the similarity measure that best matches the template graph.

DISCUSSION AND CONCLUSION

Analyzing and visualizing the multidimensional and temporal social network was demanding. Searching subgraphs based on approximate matching is difficult, especially when the graph is complex and nodes and edges hold multiple parameters. The graph given was composed of several layers that should be matched together, and we had no algorithm available to perform an automatic search and matching. Basic graph visualization approaches also did not scale to this challenge as the overall graph was large and multi-layered. Even if we had managed to visualize the whole structure, subgraphs would not have been detectable in the large graph using vision alone.

To solve the challenge, we had to combine analytical and visual methods, making the challenge a perfect target for visual analytics. Our approach was to use analytical methods to narrow down the larger problem to smaller subgraphs that could be visualized. We had hoped to confirm the final solution both visually and analytically but ended up relying more on our visual solutions because we found the graphlets to be less effective to confirm our findings than we had hoped.

We learned, somewhat to our surprise, that the graphlet matching approach was not very accurate for graph matching, especially for Task 2. Although we tried to improve the graphlet matching by weighting different graphlets according to their inverse frequency—the least represented graphlets being most informative about particular structures—the weighted profiles matching were still not entirely satisfactory. For Task 1, it was hard to see if there was a good subgraph match, while for Task 2, the graphlet similarity measure produced subgraphs of low quality. More research is needed to understand if this inaccuracy is intrinsic to graphlet-based methods or sensitive to the size and structure of the graph.

We also found it difficult to find the right similarity measures and channels to use because what worked for Task 1 was different from what helped us to find the solution to Task 2. We tried an automatic matching approach based on a greedy algorithm. However, we had trouble balancing those measures or determining which one was the most important. One aspect of that difficulty was to find which node properties of the template were the most selective to the large graph.

The graph matching result from the greedy algorithm turned out to depend on the similarity measure used. In some cases, the extracted subgraphs were good matches relative to these similarity measures but unsatisfactory overall with major differences in communication edges or origin locations of paired nodes. A future improvement of the algorithm could be to add neighborhood consistency criteria and node similarity measures when comparing node pairs to improve the edge matching accuracy. We think the algorithm can be integrated into the MatchMaker tool to suggest matching nodes and allow users to add nodes to the subgraph in a semi-automated way.

Only by trying to match nodes manually on the large graph, we found some properties that were efficient in discriminating possible node pairs. This eventually led to our solution and then to better automated extraction strategy. From these findings, our MatchMaker tool was able to help compose a good match from a couple of initial node matches. Having these initial matches is necessary for the manual matching process: it allows extracting from the large graph a subgraph of reasonable size around the initial node matches and makes the set of possible node pairs inspected by the analyst more manageable. An automated process to find motifs of the template that are rare in the large graph would have been helpful to find such initial node matches.

Overall, the challenge forced us to use all our skills in visualization, visual analytics, and graph analysis. Even if the graphlet methods were not reliable enough yet, we hope more research will improve them. For future work, it would be interesting to experiment with our approaches to combine real-world social network graphs with many types of connections and temporal relationships. Besides, we would like to combine our approach with graph representation learning methods such as *node2vec* and *graph2vec* to solve this challenge. Analyzing and understanding the characteristics of the individual graphs allows better assessing the combined results derived from a mix of automated and visual methods. We concluded that a combination of interactive visualization tools coupled with a search using classical similarity measures remains quite effective when used by motivated practitioners.

REFERENCES

1. R. Zafarani, M. A. Abbasi, and H. Liu, *Social media mining: an introduction.* Cambridge University Press, 2014.

2. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, “Network motifs: simple building blocks of complex networks,” *Science*, vol. 298, no. 5594, pp. 824–827, 2002.
3. N. Pržulj, “Biological network comparison using graphlet degree distribution,” *Bioinformatics*, vol. 23, no. 2, pp. e177–e183, 2007.
4. A. Mukherjee, B. Liu, and N. Glance, “Spotting fake reviewer groups in consumer reviews,” in *Proceedings of the 21st International Conference on World Wide Web*. New York, NY, USA: Association for Computing Machinery, 2012, p. 191–200.
5. K. Cook and R. J. Crouser. VAST Challenge 2020. [Online]. Available: https://vast-challenge.github.io/2020/
6. R. Charbey and C. Prieur, “Stars, holes, or paths across your facebook friends: A graphlet-based characterization of many networks,” *Network Science*, vol. 7, no. 4, pp. 476–497, 2019.
7. N. Pržulj, D. G. Corneil, and I. Jurisica, “Modeling interactome: scale-free or geometric?” *Bioinformatics*, vol. 20, no. 18, pp. 3508–3515, 2004.
8. P. Ribeiro, P. Paredes, M. E. Silva, D. Aparicio, and F. Silva, “A survey on subgraph counting: concepts, algorithms and applications to network motifs and graphlets,” *arXiv preprint arXiv:1910.13011*, 2019.
9. P. Ribeiro and F. Silva, “G-tries: an efficient data structure for discovering network motifs,” in *Proceedings of the 2010 ACM symposium on applied computing*, 2010, pp. 1559–1566.
10. P. Senin, “Dynamic time warping algorithm review,” *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, vol. 855, no. 1-23, p. 40, 2008.
11. W. Meert, K. Hendrickx, and T. V. Craenendonck, “wannesm/dtaidistance v2.0.0,” https://github.com/wannesm/dtaidistance/tree/v2.0.0, Aug. 2020.
12. X. Chen, Y. Li, P. Wang, and J. Lui, “A general framework for estimating graphlet statistics via random walk,” *arXiv preprint arXiv:1603.07504*, 2016.
13. D. Vysochanskij and Y. I. Petunin, “Justification of the 3σ rule for unimodal distributions,” *Theory of Probability and Mathematical Statistics*, vol. 21, no. 25-36, 1980.
14. M. H. Loorak, C. Perin, C. G. Collins, and S. Carpendale, “Exploring the possibilities of embedding heterogeneous data attributes in familiar visualizations,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 581–590, Jan. 2017.
15. T. Rito, Z. Wang, C. M. Deane, and G. Reinert, “How threshold behaviour affects the use of subgraphs for network comparison,” *Bioinformatics*, vol. 26, no. 18, pp. i611–i617, 2010.

Natkamon Tovanich is a Ph.D. student at IRT SystemX and Université Paris-Saclay, France. His research topic is visual analytics on blockchain data. Contact him at natkamon.tovanich@irt-systemx.fr.

Alexis Pister is a Ph.D. student at Inria in Aviz team and SID team of Telecom Paris. His main research areas are information visualization and visual analytics, with a focus on graph visualisation and analysis. Contact him at alexis.pister@inria.fr.

Gaëlle Richer is currently a Postdoctoral Researcher at Inria, France in the Aviz team. Her main research areas are information visualization and visual analytics with on focus on scalable techniques. Contact her at gaelle.richer@inria.fr.

Paola Valdivia is a Postdoctoral Researcher at Telecom Paris. Her main research areas are visual analytics and information visualization with a focus on dynamic network visualization. Contact her at paola.valdivia@inria.fr.

Christophe Prieur is an associate professor in the social sciences department of Telecom Paris. His main research areas are computational social science, with a focus on networks and how people manage their relationships, and sociology of data. Contact him at cprieur@enst.fr.

Jean-Daniel Fekete is the scientific leader of the Inria project team Aviz. His main research areas are visual analytics, information visualization and human-computer interaction. Contact him at jean-daniel.fekete@inria.fr.

Petra Isenberg is a research scientist at Inria, France in the Aviz team. Her main research areas are information visualization and visual analytics with a focus on novel display devices, interaction, and evaluation. Contact her at petra.isenberg@inria.fr.