Criminal Network Detection in Social Networks

IEEE VAST 2009 Challenge Analysis

Student: Ashutosh Bhandari
Course: CEG7560 - Visualization and Image Processing for Cyber Security
Professor: Thomas Wischgoll
Date: December 10 2025

Introduction

For the first time I reviewed this project I was nervous. The project involved a dataset with more than 6,000 individuals and about 29,000 connections to each other, the goal was to identify an Unknown Criminal Espionage Network hidden within all of these connections. Finding this 'criminal espionage network' among all this data felt really difficult task. In fact, I had an understanding of what the 'criminal espionage network' was, however having no clue at first as to how I was supposed to identify this network was a major obstacle to my analysis.

The project is a spin-off of the IEEE VAST 2009 Challenge, which contains benchmark data for researchers to develop and evaluate new methodologies for network analysis. The actual 'crimes' or data points, however, are much more interesting; the scenario is based around the notion that there is an insider at some organization communicating with three hustlers. Those hustlers then contact one of two 'middlemen', who in the scenario is termed "Boris", or three of them. Ultimately, every communication originates from the same source who is described in the scenario as the "Fearless Leader" of the organization. The greatest challenge to the project was not just locating the three 'hustlers', the two 'middlemen' and the 'Fearless Leader', but also verifying that the relationships made sense and ultimately allowed me to develop a solid understanding of how and why this network existed.

I had to establish if everyone involved lived in the designated cities based upon our analysis, and thus had the appropriate quantity of results measured through connection counts and to finally affirm whether the network was a linear or Boolean structure based upon some random prior connections between multiple individuals to the same person and in many cases, several individuals. I eventually became comfortable with how to handle this level of network analysis through improvements in software techniques and the collection of compliant documents that meet the challenges required.

How I started?

Before jumping into code, I spent some time thinking about what makes someone suspicious in a network like this. The project description gave some hints about expected characteristics, but I

had to translate those into something measurable. For the employee, the specs mentioned around 40 contacts on this Flitter social network. That's specific enough to work with. But here's the thing, in a network of 6,000 people, how many people have around 40 contacts? Turns out, quite a few. I found 65 candidates just in that range. So, degree alone wasn't going to cut it. The handlers were supposed to have between 30 and 40 contacts each. Again, that's a pretty common range.

The real distinguishing feature would be that all three handlers somehow connect to the employee and then funnel information through to a middleman. Boris is a go-between, representing the connection between the handler and the leader. The network viewpoint would define him having high degree of Betweenness Centrality, since he is connected to many people within that same network and most of the shortest path in the network would run through him. His role as a "bottleneck" lends itself well to someone trying to control the flow of information. Fearless Leaders need to connect with far greater numbers, probably around 100 or so.

There should be a need for an international connection, with the majority of the Fearless Leader's contacts being located outside of their city. The Fearless Leader is supposed to be leading a large organization, thus requiring a greater level of connectivity in their network footprint.

Technical Approach

I used python for this project as it is my go-to programming tool and the network analysis library NetworkX is really good for this kind of work. For handling data files, I used pandas and for visualization I used matplotlib as I already had hands on experience on it.

Loading the data was challenging for me. The dataset comes separated with tab text files with header rows that needed to be skipped. I was straight forward and I made sure the IDs were treated as integers and not strings, or the graph construction would fail later.

Building the graph was straightforward once I had the data cleaned up. NetworkX makes it easy, I just needed to add nodes and edges, and it also handles the internal representation. I added attributes to each node like their name, city, and entity type, which came in handy later for filtering and reporting.

The harder part was calculating the metrics. I needed degree centrality, which is easy it's just how many connections someone has. But betweenness centrality is computationally expensive. For a network this size, the exact calculation was taking forever. I'm talking over two minutes before I gave up and killed the process.

I researched and found that NetworkX helps to approximate betweenness centrality calculation through sampling. I can sample a fraction of total paths, instead of calculating all the shortest paths between all the nodes. By sampling 500 nodes, the estimated time to compute this information was reduced to about 10 seconds. This estimation is sufficient for ranking purposes as I didn't require precise results.

Detection Logic

Here's where things got interesting. I had to build a detection algorithm that could find these network structures. My approach was to work backwards from what I knew. First, identify employee candidates. I filtered nodes with 35 to 45 connections and scored them based on how close they were to 40. I gave bonus points if they were located in large cities like Koul, Kouvnic, or Prounov, since that matched the expected geographic pattern. I ended up with a ranked list of 65 candidates. For each employee candidate, I then looked at their neighbors to find potential handlers. This is where it gets tricky.

The handlers need to have the right number of connections (25-45 range), but they need to connect properly to a middleman. I couldn't just pick any three neighbors and call them handlers. I tried all combinations of three neighbors who fit the handler profile and checked if they shared any common neighbors besides the employee. Those shared neighbors are potential middlemen.

For Scenario A, all three handlers need to connect to the same middleman. For Scenario B, each handler connects to their own middleman, but all three middlemen must then connect to the same leader. My initial results were weird. The algorithm kept finding patterns where the employee and Boris were the same person. That didn't make sense, you can't be your own middleman. Topologically, it was creating a star pattern instead of a chain. After looking at the results more carefully, I realized I needed to explicitly exclude self-referential patterns. Once I added that constraint, the results got much better.

The leader detection was another challenge. My first attempt just looked for anyone with over 50 connections. But that seemed too lenient. When I looked at the degree distribution of the network, there are some super-connected nodes with 300+ connections. Those are the real hubs. I decided to restrict leader candidates to the top two percent of the network by either degree or betweenness centrality. This made sure I was only considering truly prominent nodes, not just moderately well-connected people.

Results

After running the analysis on the top five employee candidates, I found several valid patterns for Scenario A. The highest scoring configuration had a confidence score of 104 out of a possible 115 points, which is pretty good. The detected network looks like this. The employee is someone named @lafouge who has exactly 40 connections and is located in Prounov, which is one of the large cities. Around Koul are the three Handle people on the Handle line are handle names of @krintz and 35 others and 21 middleman's name of all three handlers name is @formenti.

The number of all the connection counts of the three handlers is the expected name. For the middleman the @clement's connection count of 21 fits the profile. This is lower than the handlers, which makes sense because Boris isn't meant to be a major hub, just a bridge. What's important is that Boris has high enough betweenness centrality to act as a connector between

the handlers and the leader. The Fearless Leader is @dykema. This is where the results get really convincing. @dykema has 386 connections, making them one of the top nodes in the entire network. They rank third overall by degree and have connections spanning 248 different cities. If you're looking for someone who could run an international criminal organization, this profile fits perfectly.

There is a strong correlation of 83.3% between geographic patterns of Peer Networks and actual geographic locations of all members of a peer network. The majority of OG members can be found within their expected geographic market type. As anticipated, all employees/handlers can be found in Large City Locations, as well, while the OG Leader can be located in what would be expected to be a very large Distribution/Hub Center. The only exception is Boris, who also resides in the Large City. All other specified Locations for OG Members comply with the idealized configuration. I also attempted to locate possible Scenario B patterns that depicted multiple middlemen linked to a single leader.

Unfortunately, I did not identify any Scenario B patterns that attained confidence interval scores sufficient to provide me with confidence in my conclusions. Instead, the evidence supports the theory that a greater number of middlemen exist in the same distribution area in this data source and most closely resemble the Scenario A pattern for distribution systems. Either that, or my detection algorithm needs more tuning for the three-middlemen case. Given the time constraints, I focused on making Scenario A detection as good as possible.

Visualizations

I created four main visualizations for this project. The first is a full network overview showing all 6,000+ nodes. You can see there are some very dense clusters and a lot of sparse connections. The degree distribution plot shows that most people have very few connections, maybe 5 or 6, while a small number of people have hundreds. It's a classic power law distribution, which is typical for social networks.

The centrality comparison chart shows the top 20 nodes by four different metrics degree, betweenness, closeness, and clustering coefficient. What's interesting is that @dykema, the detected leader, appears in the top rankings for multiple metrics. That cross-validation gives me more confidence that the detection is picking up on something real rather than noise.

The criminal network visualization shows just the detected network members and their immediate neighbors. I color-coded each role, the employee is red, handlers are orange, Boris is gold, and the leader is dark red. You can see the chain structure pretty clearly in the layout. The employee connects to the handlers, the handlers all connect to Boris, and Boris connects to the leader. There are other connections too, which is expected since these people have other contacts, but the main pathway is visible.

Challenges and Limitations

The biggest technical challenge was the computation time for betweenness centrality. An approximation method was required; however, it adds inaccuracy to the results. All of the relative rankings should still be valid, but the absolute values may not be accurate. Given the requirement of running this project in a reasonable amount of time, the trade-off between inaccuracy and running time was worthwhile. Another challenge was deciding on the threshold values. How strict should I be about the expected number of connections? If I narrow the range too much, I might prevent seeing the actual network; if I widen the range too much, I will see numerous with false positives.

As a result, I have slightly widened the range of each category (for example, I used 25-45 for handlers versus 30-40) to capture variabilities in the data. Real networks don't follow exact specifications. The self-referential pattern issue was frustrating because my initial code was technically correct, it was finding high-scoring configurations. But those configurations didn't make logical sense. I learned that I cannot just rely on scoring function as my primary method of determining what topology is valid. This serves as a reminder to use more than one approach when solving optimization problems rather than relying solely on the algorithm to determine the optimal option based only on the scoring function.

With respect to the detection of Scenario B, I had difficulty finding consistent results after having tried modifying the parameters and the scoring weights. One issue may be due to the vastness of the combinatorial search space; even attempting to evaluate the combinations of three handlers and three middlemen would rapidly become an expensive operation. To limit the overall search space to just the top 10 prospective handlers for each employee, I chose this method and successfully narrowed my search down to something manageable, but I may also have missed some potential patterns.

What I Learned

The most important thing that I learned is to convert my theoretical knowledge to practical. In the research paper, it might be very straightforward to write out the steps that the algorithm follows. But, in practice, I found that computing power limitations and data quality issues, as well as unexpected edge cases, and can arise when implementing these algorithms.

I have found that in few instances that when conducting extensive network analysis, temporal approximations and heuristics must be accepted as part of the overall process of a large-scale network system; using an exact algorithm, while very appealing, may take too long to complete up to many hours. Although the technique that I have used sacrifices some accuracy, it provides a large increase in processing speed. For most cases, that's the preferred compromise. I found that it was very important to validate my findings by different means, as just because a pattern was discovered, it does not necessarily indicate that there is a meaningful connection between the two variables.

I also had to check geographic consistency, topological structure, and statistical significance. The configs that passed all the check were much more important than the ones that just had high scores. The visualization aspect was more important than I expected. When I could see the network structure laid out, it helped me spot issues with my algorithm. For example, the self-referential patterns were visually obvious as star topologies, even though they had good numerical scores. Being able to visualize what the algorithm was doing helped me debug and improve it.

Future Work

If I had more time, I would definitely improve the Scenario B detection. The current algorithm is the same as Scenario A but with an extra layer of middlemen. I think a better approach would be to use different scoring weights or a different search strategy for the three-middlemen case.

I'd like to explore community detection more thoroughly. NetworkX has a Louvain method implementation that finds clusters in the network. I calculated communities but didn't use them much in the detection logic. It would be interesting to see if the criminal network forms its own community or if it's spread across multiple communities.

Another interesting extension would be temporal analysis. The dataset is static, but in reality, social networks change over time. People make new connections and drop old ones. The ability to review the development of the criminal network over several different periods would allow me to identify trends and influence patterns between those periods. This may show some information that is not obtainable through only one point of view.

I'd want to validate these results against ground truth if it exists. The IEEE VAST Challenge probably has known answers for testing purposes. Comparing my detections to the planted criminal network would tell me how accurate the algorithm really is. Without that validation, I can only say that I found plausible patterns, not that I found the correct ones.

Final Thoughts

Building algorithm taught me so much that I didn’t expect I would get this involved on this project. What seemed like a straightforward pattern matching problem turned out to require careful algorithm design, parameter tuning, constraint validation, and multiple rounds of debugging.

The detected criminal network with @lafouge as the employee and @dykema as the leader seems credible based on the metrics I looked at. The confidence score of 104 indicates a strong match to the expected pattern. The leader has 386 connections spanning 248 cities, which definitely fits the profile of someone running a large organization. All the roles are topologically distinct with no self-referential loops, and the geographic distribution makes sense.

Would I bet money that this is the criminal network? No, not with external validation. But based on the data I have and the analysis techniques I used this is the most likely configuration. If I were a real professional, this is where I'd focus my investigation.

The code I wrote is pretty modular and could be applied to other network analysis problems. The approach of scoring configurations based on multiple criteria and then ranking them is generalizable. The visualizations look professional, and the report captures what I did and why I did it. Not bad for a semester project.

Technical Specifications

To create the same result, here are the key values and parameter I used:

The employee range was 35-45 connections with a scoring preference for exactly 40. Handler range was 25-45 connections. Boris had to have between 15-80 connections with betweenness centrality above 0.001. Atleast 50 connections were needed for the leader and had to be in the top two percent of the network by degree or betweenness.

The software used for conducting graph operations was NetworkX 3.2.1, while pandas version 2.1.4 was employed for data processing, with the Community Detection package (python-louvain v0.16) utilized for conducting community detection analyses via Louvain methods. The k parameter used for approximating the betweenness centrality measure in relation to the number of samples was set to 500. Visualizations were produced at 300 DPI resolution to provide report-quality graphics**.**