Investigative graph search using graph databases

Shashika R. Muramudalige, Benjamin W. K. Hung, Anura P. Jayasumana, Indrakshi Ray
Department of Electrical and Computer Engineering
Colorado State University
Fort Collins, Colorado 80521, USA
{shashika.muramudalige, benjamin.hung, anura.jayasumana, indrakshi.ray}@colostate.edu

***Abstract*—Identification and tracking of individuals or groups perpetrating latent or emergent behaviors are significant in homeland security, cyber security, behavioral health, and consumer analytics. Graphs provide an effective formal mechanism to capture the relationships among individuals of interest as well as their behavior patterns. Graph databases, developed recently, serve as convenient data stores for such complex graphs and allow efficient retrievals via high-level libraries and the ability to implement custom queries. We introduce PINGS (Procedures for Investigative Graph Search) a graph database library of procedures for investigative search. We develop an inexact graph pattern matching technique and scoring mechanism within the database as custom procedures to identify latent behavioral patterns of individuals. It addresses, among other things, sub-graph isomorphism, an NP-hard problem, via an investigative search in graph databases. We demonstrate the capability of detecting such individuals and groups meeting query criteria using two data sets, a synthetically generated radicalization dataset and a publicly available crime dataset.**

***Index Terms*—sub-graph isomorphism, graph pattern matching, similarity measure, graph databases**

I. Introduction

Exploration of social networking and behavioral data with the goal of identifying certain latent and emergent behaviors of individuals and groups is necessary in domains such as homeland security, cybersecurity, consumer analytics, and behavioral health. Such behavior patterns in many cases are naturally expressed in the form of graphs [1], and often observable in interactions via social networks [2]. In cybersecurity, organizations continually seek to prevent threats by detecting risk potential using performance-related and technical indicators recorded over time [3]. Businesses are interested in an individual's online activities and purchases over time to track on the customer and determine the potential for future purchases via consumer analytics [4] [5]. In behavioral health, identification of precursors to suicide over time is of vital interest [6] [7]. Graph-based models, mining, and tracking algorithms provide a powerful approach for these problem domains.

Of specific interest to our work is homegrown violent extremism, a major threat spreading to many parts of the world [8]. An individual's transformation towards violent radicalization typically involves a sequence of steps; examples of such steps include the consumption of extremist ideas and propaganda through internet sources and the immersion with other radicalized peer groups [1] [8] [9]. It is a daunting challenge to detect the emergence of indicators among individuals in a large population. In many cases preparatory tasks, or even attacks, have been carried out as groups. Detection of such cases is extremely challenging as individual behaviors may not follow all the profile components or steps, while the group as a whole does. Thus, efficient mechanisms are also necessary to track the partial matching profiles of individuals which taken together satisfies more complete version of the profile of interest.

The focus on this study is inexact pattern matching techniques to identify the behavior of suspicious persons and groups. As the examples above indicate, the techniques are applicable to many other domains. The behavior of extremists and criminals can be descriptively studied through their past records and activities. Radicalization trajectories of homegrown jihadists based on 335 known American jihadists are characterized in [8] in the form of 135 detailed forensic biographies that detail their pathways. Probabilistic models of radicalized pathways are presented in [1].

Analysis of crime data for prediction and prevention often also involves similar mining and analysis. Radicalization as well as criminal data are highly connected and can contain knowledge of the interpersonal and/or social media connections among individuals as well as association with suspicious activities, events, and locations. Such data representation implicitly forms a complex multi-dimensional network, and requires complex graph search and pattern matching operations to facilitate deeper analysis. Some online social network platforms also offer graph search on their networks [10] [11]; but they allow limited amount of data due to confidentiality or legal constraints. Data integrity is a concern in online social networks due to fake accounts created to purposely mislead. Data integrity is one of the key dependencies of investigative search; the biographical data used in this study was manually collected from court documents and other public sources on American homegrown Salafi-jihadist terrorism offenders [8] [1].

Graph databases are indispensable for pattern-based querying over large volumes of data that are characteristic of our

This work was supported by the U.S. Department of Justice, Office of Justice Programs/National Institute of Justice under Award 2013-ZA-BX-0005. Opinions or points of view expressed in this article are those of the authors and do not necessarily reflect the official position of policies of the U.S. Department of Justice.

978-1-7281-4129-9/19/$31.00 ©2019 IEEE
DOI 10.1109/GC46384.2019.00017

problem domain. Unlike SQL or NoSQL databases, they are designed to treat the relationships between data as equally important as the data itself [12]. The nodes and edges are considered as separate data structures in the database. Graph databases reshape graph problems and address them through novel approaches. Neo4j [13], a prominent graph database available in the market, also includes basic graph traversal algorithms and the capability for users to implement custom procedures and extensions to meet unique requirements. The developer libraries simplify the access of graph data structures and the implementation of custom query procedures. Although the focus of this work is graph analytics for querying, the instantiation of data in the Neo4j graph database also enables further graph theoretical analysis using a rich, built-in library of parallel versions of graph algorithms optimized for the database [14] [15]. Algorithms for centrality, community detection, and path finding are all readily available for employment on the data that will further deepen the analytic insights for investigators.

In this study, we use a Neo4j graph database as the data storage and implement an algorithm to fetch profiles of interest from the graph database. The algorithm is based on sub-graph pattern matching using a scoring mechanism. It is developed as a set of Neo4j procedures called PINGS (Procedures for Investigative Graph Search) where it works as custom query into the database. In graph theory, this problem is named as the sub-graph isomorphism, which is known to be an NP-hard problem [16]. Two data sets are used to validate and evaluate the proposed algorithm. One is a synthetically generated radicalization dataset (RD) [17], mimicking the behavioral models of homegrown radicalized extremists [1] [8], that is used to evaluate the scalability of the method. A query graph is defined to represent the generic behavior of an extremist, criminal or their group. So, the query graph represents a superset of potential behaviors of suspects, and the score mechanism calculates the similarity between each individual or group and the query graph. The similarity score is a measurement of the similarity between an individual behavior profile and the set of behaviors exhibited by an extremist. Second data set is a crime dataset (CD) accessible via the Neo4j sandbox [18]. It contains the street crime data in Greater Manchester, UK. The dataset was largely extracted from public sources about the prevalence, location, and type of crime [19], but does not include any real information about persons related to the crimes. The dataset was enriched to include fake criminals, investigators, and timestamps for demonstration purposes. The proposed algorithm was applied to this crime dataset to search for similar crime patterns with respect to a criminal or a specific location. The study shows the accuracy of the results on several investigative domains; radicalization and crime pattern detection. Moreover, the proposed approach is sufficiently scalable to manipulate hundred thousands of individuals and the query performance depicts that it is capable of retrieving individuals or groups within a reasonable time frame.

The contribution of this paper is as follows: a) We propose an investigative graph search technique using graph databases. The proposed algorithm is capable of retrieving individuals and groups for further investigation based upon their similarity to a query graph. This is a reliable and scalable investigative graph search approach compared to the *Investigative Simulation* [20]. A library of routines useful for investigative graph search using Neo4j database is presented. b) We propose a novel mechanism to detect groups of individuals that collectively satisfy the query graph. The motivation for such an algorithm is the observation that groups of people in a conspiracy have been known to cumulatively perform the set of suspicious activities in the behavioral model (query graph).

Section II reviews some related work. Section III presents the preliminaries and notations, data schema, node categorization on graph databases, and proposed algorithm. Section IV depicts the experiments and results. Section V concludes the paper and enumerates future research directions.

II. Related Work

Our study builds on the inexact sub graph isomorphism problem. Inexact matching involves finding subgraphs that closely match a query graph. Exact match solutions do not exist for many problems involving social networking data, e.g., during radicalization process or when data is incomplete. Exact subgraph match is relatively less complex as it looks only for exact matches. Ullmann's Algorithm [21] is the first method to find isomorphic patterns of query graphs in large graphs [16]. Tree search based algorithms are relatively efficient for exact matching related to big-data problems [22] [23] while optimizing the memory consumption.

BB-Graph [16] is a branch-and-bound technique for exact subgraph isomorphism for querying in graph databases. They also use the Neo4j graph database. A top-k pattern matching technique is proposed in [24] based on two criteria for ranking the matches, relevance and distance functions. Then the generalized top-k matching function is introduced as a combination of both generalized relevance and distance functions. *Dual simulation* is an enhancement of Ullmann's algorithm which searches for binary match relations between query and data graphs. It preserves not only parent-child relationships, but also child-parent relationships in the match and thus produces more sensible matches [25].

Investigative Simulation [20] is an extension of dual simulation to achieve inexact matching in isomorphic patterns to produce more sensible matches of potential subjects for further investigation. They also propose a roadmap of further advancements necessary for investigative search, particularly for detecting the radicalization of homegrown violent extremists based upon online and offline behavior. They introduce a categorical node labeling mechanism for investigations by giving weight to each node based on the activity represented by the node. An investigative framework for detecting radicalization trajectories in large heterogeneous graphs [26] demonstrates the scalability of the approach through a large dataset. Investigative search is applicable in other domains like consumer analytics, behavioral health, and cyber security [27].

Our work builds on [20] [26] [27] and provides a more scalable investigative graph search via graph databases. In [20] [26] [27] the data is stored in files and evaluations are done as post processing steps after the data are fed into the program. However, this is extremely inefficient as data is dynamically updated, and it is necessary to take into account impact of data storage for investigative graph search. In our study, we focus not only on maintaining a graph data store but also leveraging the features of the graph database to fetch investigative results as a custom query in real time.

III. Investigative search of graph databases

Terminology and notation is presented here before discussing the proposed algorithm and the exertion of some features of graph databases. Mostly, investigative graphs follow the same graph structure; persons bind with set of indicators/activities.

Definition 1. Investigative graph

An investigative graph is defined as a heterogeneous directed graph $G = (V, E, L, R)$, where

- V is a finite set of nodes, n denotes the cardinality of V, that is, $n = |V|$
- $E \subseteq V \times V$, where (v, v') is an edge from v to v'
- L denotes the label set of nodes in the graph, for example, $L = \{Person, Indicator, Social\_media\_acc, Post, ...\}$
- R is the set of edge types, for example, $R = \{exhibits, knows, posts, ...\}$

The Data graph is given by $G = (V_G, E_G, L_G, R_G)$, and the query graphs by $Q = (V_Q, E_Q, L_Q, R_Q)$

Definition 2. Match node

There are 2 ways to match a node based on the requirement stated in Q. Some nodes only need to be matched with the label and some nodes require to be matched with node properties.

- *Label match*
  if $u \in V_Q$, $v \in V_G$ and $l'(u) = l(v)$ where $l' \in L_Q$ and $l \in L_G$, then labels of u & v nodes are matched
- *Properties match*
  If P_V denotes the properties set in node v and P_U denotes the properties set in node u
  For given $p \in P_V$, $p' \in P_U$ and $p'(u) = p(v)$, then properties of u & v nodes are matched

Definition 3. Match edge

The direction and the relationship type should be matched to consider the edge similarity.
For an edge $e_u = (u, u') \in Q$ and there exists $e_v = (v, v') \in G$ and $r'(e_u) = r(e_v)$ where $r' \in R_Q$, $r \in R_G$, then e_u & e_v edges are matched

We implement similarity measures as Neo4j procedures to detect suspicious individuals (*similarityMeasure*) and groups (*neighborhoodMeasure*). These procedures generate complex custom queries to the graph database. Procedures are changed for crime dataset (*crimeAnalysis*) to gain the compatibility and the dynamicity for crime pattern detection without changing the core algorithm.

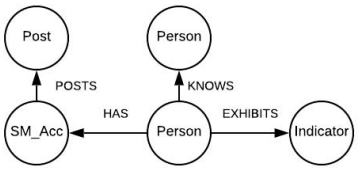

Figure 1. The data schema of radicalization data

Table I
NODE TYPES IN RADICALIZATION DATASET (RD)

Node type	Node name	Description
QF (Query Focus)	• Person	The initial node represents the subject where the search starts
IIRA (Individually innocuous)	• Purchase Weapons	Behavior that is not harmful individually but if IND or RF indicators are present, then IIRA activity could be a potential threat
IND (Indicator)	• Received Training • Referred Radicalized Materials • Suspicious Travel	A behavior that should be traced further
RF (Red Flag)	• Carried an attack • Detonated a bomb	A high risk behavior that should be further investigated immediately
SM	• Social Media Account	Represents a social media account of a Person
EP	• Extremist Post	Represents a post which contains radicalization related content.

A. Radicalization data schema

The data schema (Fig. 1) represents the preliminary behavior of a radicalized individual. Person denotes the subject of the network and connects with other persons via *KNOWS* relationship. Indicator depicts a suspicious activity and links with *EXHIBITS* relationship. A radicalized person can have multiple Social Media Accounts (*SM_Acc*) to disseminate his interests and radicalized ideologies. Those accounts link with *HAS* relationship and such posts bind with *POSTS* relationship.

B. Node categorization for indicators

The node categorization is performed w.r.t. the data schema and the characterization given in [20]. Radicalized activities (Indicators) are classified into 3 classes based on the severity. Table I shows the node types including the indicator classification. Crime dataset contains various crime types which we categorize into two classes as indicated in Table II.

Figure 2 shows an example of a query graph for radicalization dataset. A complete match with the query graph, in our case, is a behavioral indicator that identifies the radicalization. In effect, the query details all the indicators one could expect from a violent extremist. By querying the database for this pattern, we are searching for individuals who have exhibited all or some of these indicators as means to identify and prioritize high-risk individuals. We calculate similarity score for different subgraphs in the database with respect to the

Table II
NODE TYPES IN CRIME DATASET (CD)

Node type	Crime Type	Description
RF (Red flag)	• Possession of weapons • Criminal damage and arson • Violence and sexual offences • Drugs • Burglary	High risk crimes that law enforcement need to prioritize for taking actions straight away
IND Indicator	• Public order • Theft from the person • Other theft • Vehicle crime • Robbery • Other crime • Shoplifting • Bicycle theft	Comparatively less severe crimes than RF, but still need further investigating to locate criminals.

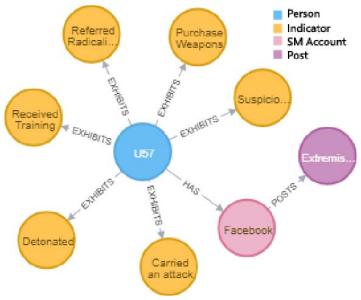

Figure 2. A query graph for radicalization dataset (RD)

query graph and fetch matching sub graphs with similarity scores above a threshold value.

C. Similarity measure for radicalization detection

The Neo4j procedure consists 4 input parameters as follows.

$$similarityMeasure(similarityThreshold, redFlagMultiple, queryLabel, queryFocusLabel) \quad (1)$$

- *similarityThreshold* – A similarity score is calculated for each user based on his activities. It is normalized to range (0,1). The query graph score *similarityThreshold* is used to identify matching subgraphs.
- *redFlagMultiple* – This attribute is used to highlight the highest risk radicalized activities [20]. The *redFlagMultiple* (≥ 1) is used to multiply (weight) high risk activities as a weighted (multiplicative) value. The end result is a prioritization of sub graphs by similarity score due to the presence of high risk activities.
- *queryLabel* – A node in a Neo4j graph database can have multiple labels. This allows us to identify a query graph within the database by appending another label (eg: 'Query'). Therefore, the query graphs can be easily updated for different experiments which the algorithm picks up within the database itself.
- *queryFocusLabel* – The node label indicates the starting point of the algorithm. In our context, we have to provide the label of the person-'User'. Then the algorithm scans all the persons and evaluates the similarity measure for each person.

e.g., *similarityMeasure(0.7, 2, 'Query', 'User')*

Algorithm 1: *similarityMeasure*

inputs : Q : Query graph with n vertices
S : Similarity threshold
W : Red-flag multiple
output: M : Set of matched graphs

$M \leftarrow \emptyset$
$C \leftarrow getConfigurationList$
Get the QF node (Q_{qf}) from Q
$S_{qf} \leftarrow \{s' \mid s' \in \mathcal{V}_G \,\&\, matchNode(Q_{qf}, s')\}$
foreach $s' \in \mathcal{S}_{qf}$ **do**
  $M_G \leftarrow searchSimilarGraphs(s', Q, W, C)$
  $S_M \leftarrow getMatchedScore(M_G)$
  if $S_M \geqslant S$ **then**
    Add M_G to M
return M;

Algorithm 2: *searchSimilarGraphs*

inputs : s' : QF node in V_G
Q : Query graph
W : Red-flag multiple
C : Configuration list
output: N : All matched nodes
A : Matched activity nodes
S : Matched similarity score

$D_M \leftarrow s'$
foreach $q' \in \mathcal{Q}$ **do**
  $M_N \leftarrow \emptyset$
  foreach $s' \in \mathcal{D}_M$ **do**
    if $matchNode(s', q', W, C)$ **then**
      Remove s' from D_M
      R_Q = $\{r_q \mid r_q \in \mathcal{Q}_E \,\&\, outgoingEdges(q')\}$
      foreach $r_q \in \mathcal{R}_Q$ **do**
        R_G = $\{r_g \mid r_g \in \mathcal{G}_E \,\&\, outgoingEdges(s')\}$
        foreach $r_g \in \mathcal{R}_G$ **do**
          m_n = $matchEdge(r_q, r_g)$
          if m_n != *null* **then**
            $M_N \leftarrow m_n$
  $D_M \leftarrow M_N$
return N, A, S;

D. Neighborhood measure for radicalization detection

$$neighborhoodMeasure(similarityThreshold, redFlagMultiple, queryLabel, queryFocusLabel) \quad (2)$$

In some cases, radicalized individuals are known to work in a conspiracy with individuals or groups with similar interests for specific tasks. Therefore, finding single-person subgraphs by calculating the individual similarity scores is not sufficient to detect such collectively suspicious behavior where the indicators are effectively dispersed among a group of individuals

[26] [27]. We propose a neighborhood measure to detect such group activity. The procedure has same input parameters as the *similarityMeasure* but it internally calls algorithm 3 to detect suspicious groups.

E. Similarity measure for crime pattern detection

The nature and structure of crime data is different from that of radicalization dataset. We look for different crime patterns based upon a criminal or a location. The procedure parameters are altered to address our requirement while deploying the same algorithm. The definition of the modified procedure is as follows.

$$crimeAnalysis(entity, identifier, identifierValue, relationshipToCrime, redFlagMultiple, similarityThreshold) \quad (3)$$

When compared to radicalization detection that utilized a canonical violent extremist pattern, we enable much more flexibility in the crime pattern analysis by enabling analysts to select a particular crime pattern within the data as a query graph. The procedure then searches for the existence of similar patterns (by criminal or location) throughout the database to aid in the identification of others perpetrating a similar set of crimes or locations that are subjected to the same types of crimes. The first 3 inputs define the root node for analysis. *Entity* represents the node label (in this dataset the *Location* or the *Person*). *Identifier* means the unique property of the node to be recognized and *identifierValue* depicts the particular property value. It is also essential to provide the relationship type to the crime node represented with the parameter *relationshipToCrime*. *redFlagMultiple* highlights serious crimes based on the table II. *similarityThreshold* serves the same function as before. Criminal pattern detection is carried out by, e.g., *crimeAnalysis('Person','nhs_no','444-91-2379', 'PARTY_TO', 0.8)*
while crime location pattern detection, is carried out by invoking, e.g., *crimeAnalysis('Location', 'postcode', 'M5 3WT', 'OCCURRED_AT', 0.9)*

F. Similarity measure

Algorithm 1 describes the proposed similarity measure. Initially, the algorithm filters out all root user nodes (line 4). Then for each user node, it calls the *searchSimilarGraphs* function (line 6). Neo4j graph database allows adding multiple labels or properties to any node or an edge. So, we maintain a configuration list (C) which is unique to a particular data schema, includes labels and properties of nodes which need to be matched through the algorithm. *searchSimilarGraphs* function calculates the similarity score while searching for a query graph in a data graph. If the calculated similarity score is larger than or equal to a given *similarityThreshold* that data graph is added to the result set (M).

Algorithm 2 explains the approach to search for similar subgraphs based on each user node. It consists of *matchNode* (**Def 2**) and *matchEdge* (**Def 3**) functions to match nodes and edges respectively. *matchEdge* has another feature that

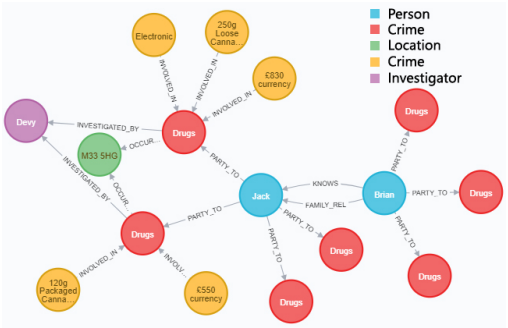

Figure 3. Identify drug network using neighborhood measure (CD)

matchNode includes inside the *matchEdge* function to match the end node. If an edge matches, it implies that both in and out nodes are also matched. Moreover, *matchNode* and *matchEdge* work based on the metadata in the configuration list (C). The algorithm maintains a queue data structure to match data nodes and store potential data nodes. Based on the query graph, it only searches for respective potential nodes in the data graph. Potential data nodes are identified according to the outgoing edges of any matched node. Then these nodes are put into a queue and considered as potential matches with query nodes. This approach helps to avoid unnecessary traversal through the whole data graph.

G. Neighborhood measure

Algorithm 3 explains the proposed neighborhood measure. This is introduced as an aggregating schema to measure collective indicators exhibited by a particular person and his/her known associates. Before calling this function, it retrieves the connected user group (neighbors) for a particular user and takes the neighbors set as an input parameter(N). Then it performs the *searchSimilarGraphs* function (Algorithm 2) for each neighbor to get matched graphs (line 5). The key point here is that a neighbor should perform at least one different indicator compared to the particular user. Then only that neighbor is considered as an eligible contributor to a team/group. In that way, it checks whether each neighbor could be a potential person who contributes to achieve a set of indicators in the query graph as a group of people. *updateCollectives* function maintains collective indicators of a group and that indicator set is matched with the query graph. The *checkEligibility* function checks the neighbor's eligibility of being a contributor to a team or a group by checking he/she has performed at least an additional indicator than the particular user.

H. Example: Detecting drug networks

We provide an example of how the neighborhood measure may be helpful in uncovering group involvement in crimes. In the crime database, there are 2 charge types for drugs crimes, namely 'Possession of drugs' & 'Intent to supply'.

In this case, the query graph contains different drug charge types and only 'drugs' as the crime type. All 3 crimes related to Brian (Fig. 3) are drugs and charge type is 'Possession

Algorithm 3: *searchNeighborMatchedGraphs*

inputs : N : List of Neighbor nodes
Q : Query Graph
W : Red-flag multiple
C : Configuration List
M : Matched Graph
S : Matched Graph Score

output: N_G : Set of matched graphs

$initialCSet \leftarrow updateCollectives(Q, M)$
$activityCSet \leftarrow \emptyset$
$N_G \leftarrow \emptyset$
foreach $n' \in \mathcal{N}$ **do**
  $M_G \leftarrow searchSimilarGraphs(n', Q, W, C)$
  $nodeCSet \leftarrow updateCollectives(Q, M_G)$
  if $checkEligibility(initialCSet, nodeCSet)$ **then**
    $activityCSet \leftarrow$
     $applyCollectives(activityCSet, nodeCSet)$
    $N_G \leftarrow \mathrm{M}_G$
if $checkSimilairtyScore(activityCSet) \geqslant S$ **then**
  return N_G;

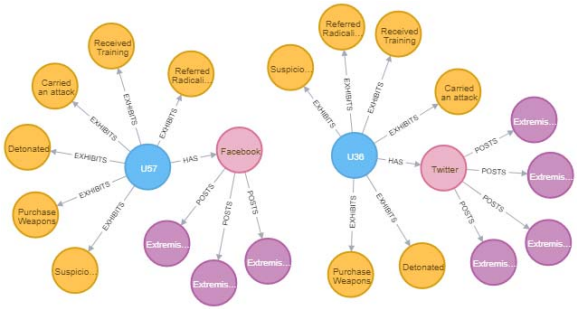

Figure 4. Similairty measure for exact match (RD)

of Drugs'. He has a family relation with Jack who has been charged for both 'Possession of drugs' and 'Intent to supply' and found with 'packaged & loose cannabis' too. So, there is a high probability that Jack is Brian's cannabis supplier. Moreover, Jack was caught twice with cannabis in a certain location (postcode – 'M33 5HG') and investigators can focus on others related to that location to trace the drug network further.

IV. Experiments and Results

Several experiments are performed in different graph database setups *similarityMeasure*, *neighborhoodMeasure* and *crimeAnalysis* procedures in PINGS library. The query performance was also evaluated across different sizes of radicalization datasets and the crime dataset.

A. Radicalization data analysis

Figure 4 depicts the results for an exact pattern match (*similarityThreshold=1* & *redFlagMultiple=1*) based upon the query graph in Figure 2. As we explained, the query graph is

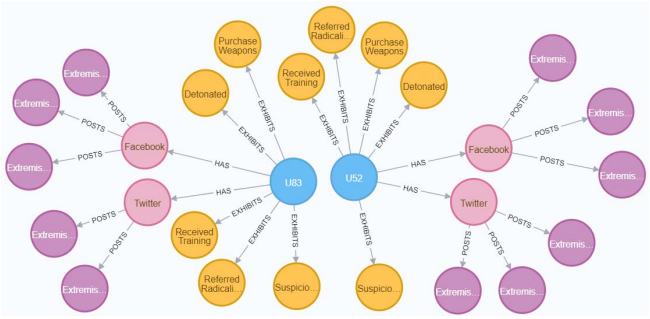

Figure 5. Similairity measure: Inexact match with similarityThreshold 0.7 (RD)

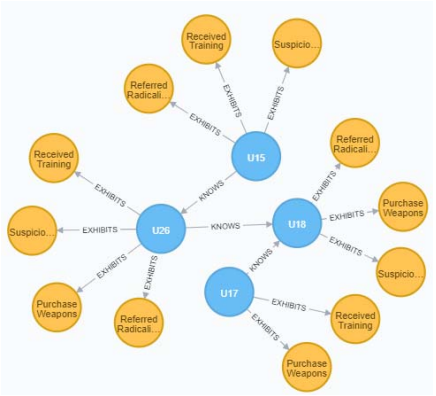

Figure 6. Neighborhood measure: Inexact match with similarity threshold 0.8 (RD)

also defined inside the dataset using different node label. So, the left graph shows the query graph with the user id 'U57' and the right graph, user 'U36' is an exact match for the query graph. Even though, the number of social media posts or social media accounts are not matched exactly, it matched with all other indicators while taking into account prioritized indicators to be matched. In short, this validates our approach because the query graph also retrieves an exact match which belongs to the data graph too.

Some of the inexact match results are depicted in Figure 5 when the *similarityThreshold* is 0.7 and *redFlagMultiple* is 1. Both persons 'U52' and 'U83' have demonstrated 5 (out of 6) indicators and both used social media accounts to propagate radicalized content. This is one of the major strengths of the approach: it is able to detect lookalike suspicious behaviors which is not exactly matched with a given query graph.

An exact match result of *neighborhoodMeasure* identifies a group that collectively exhibits all the indicators in the query graph and marks it as suspicious. It may be a team who have already committed crimes and investigators may be able to search for their other connections to eliminate future threats. Moreover, the customized Neo4j procedure allows investigators to find suspicious groups that are not exact matches with the query graph by reducing the *similarityThreshold*. Fig. 6, interprets an inexact match result (For a better visualization, the social media details were truncated);

Figure 7. Query graph for location 'OL10 2JL' (CD)

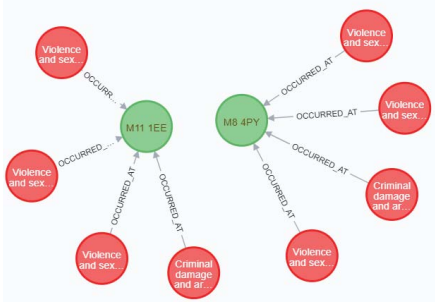

Figure 8. Exact crime patterns by location based on the query graph (CD)

all 4 persons who know each other have shown suspicious activities. 3 of them indicate 'Received training', 'Purchase weapons' and 'Suspicious travel'. It is possible that these individuals may have worked as a group and have or will perpetrate an attack in the near future. At the very least, it points to a group that warrants immediate investigation. Investigative bodies are now empowered with knowledge of potential involvement by other individuals in the group.

B. Crime location analysis

Figure 7 depicts the crime pattern for a location (postcode – 'OL10 2JL') which was retrieved as the query graph. The figure 8 shows some of the exact patterns of the other locations based upon the query graph in figure 7.

C. Criminal analysis

Fig. 9. shows the crime pattern of the criminal called 'Brian'. We just input the person's identifier and it spontaneously picks his crime details and presents his crime pattern. Figure 10 depicts the results when the similarity threshold is reduced to 0.7. The crime patterns of 'Alan', 'Kathleen' and 'Diana' are fetched as somewhat similar crime patterns to the crimes related to 'Brian'. The database also fetches their relationship (if one exists) and depicts whether they know each other. So, this querying capability could be highly important to investigators to identify other criminals with similar crime patterns, to query for others who may have similar modus operandi, as well as to trace the potential connections among those criminals. Since, we are may not be definitively interested in the order in which a criminal committed his offenses, the inexact similarity measure identifies somewhat likely crime patterns irrespective of the order because we focus on crime types and their categorization based on table II.

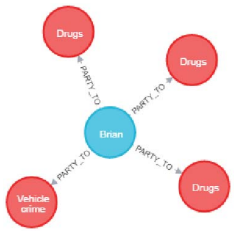

Figure 9. Query graph for person 'Brian' (CD)

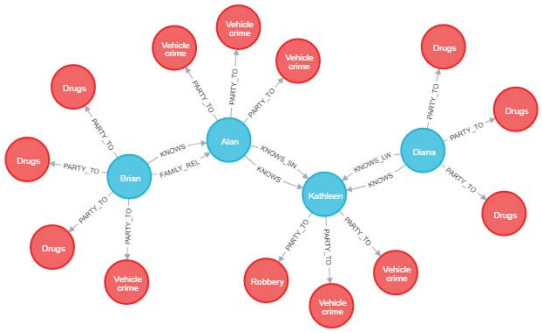

Figure 10. Inexact (similarity threshold = 0.7) crime patterns by person (CD)

We also ran some query performance tests on Neo4j graph databases. We use a machine with Intel i5 2.20GHz CPU and 8GB RAM for all our experiments. We generate different size of radicalization datasets using our data simulator. We also maintain a similar the graph density in each case where persons averaged 3 indicators. The table III depicts the details of the radicalization datasets. *similarityMeasure (SM)* and *neigborhoodMeasure (NM)* average query time for each dataset size illustrates in figure 11. It runs for 2 scenarios as exact match and inexact match (*similarityThreshold* – 0.8). *neigborhoodMeasure* takes more time because it searches all possible group combinations and *similarityMeasure* just evaluates for individuals. While the dataset size is increasing, the query time difference between the exact and inexact match for each measure is also increasing. This basically occurs because the number of group combinations to inspect in the *neigborhoodMeasure* is significantly increased with the number of the persons.

We also check the query time for the crime dataset for *crimeAnalysis* procedure by locations. The dataset consists 369 criminals, all together 61521 nodes and 105840 edges. Figure 12 interprets the results and the average inexact query time is slightly higher than the exact match queries. When the graph size (number of crimes in a graph) increase, the query time has an exponential trend after 10 crime nodes per graph.

Table III
CHARACTERISTICS OF RADICALIZATION DATASETS

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
No of persons	100	1000	10000	100000
No of nodes	979	9627	96590	966572
No of edges	909	8178	78590	784053

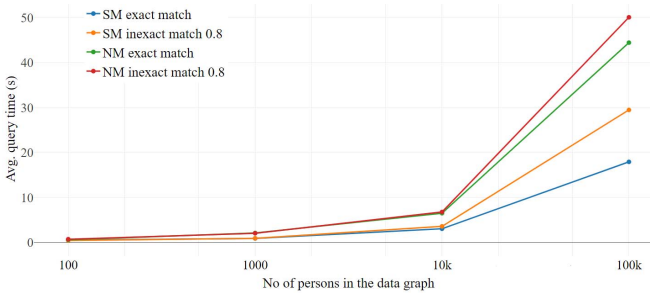

Figure 11. Avg. query time vs no of crimes involved in a graph (RD)

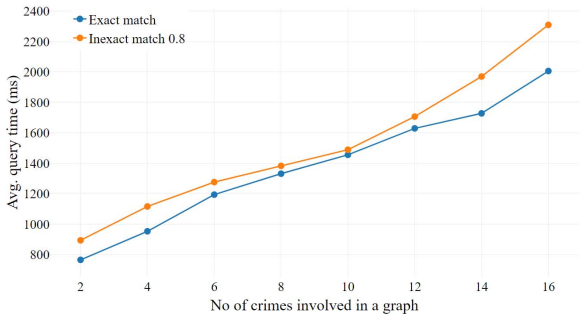

Figure 12. Avg. query time vs no of crimes involved in a graph (CD)

V. Conclusion & Future work

Investigative graph search based on inexact pattern matching was presented using graph databases. Results using a synthetically generated radicalization graph database and a real crime graph database depict the accuracy and the efficiency of the proposed investigative graph search. We show how features in graph databases can be efficiently applied for investigative use cases. A database library (PINGS) with a set of custom procedures and comprehensive details are made available in [17]. We demonstrated the capabilities of PINGS library and described its similarity scoring mechanisms to identify potential suspects, groups, and patterns.

The use of timestamps of activities to enhance the search outcomes are being developed. The complexity analysis for algorithms and multi-threaded search procedures to improve the query performance will be constructed. Moreover, the query performance in distributed large graph databases will be inspected to further scale-up the proposed investigative graph search.

Acknowledgment

The authors wish to thank Prof. J. Klausen and Western Jihadism Project group at Brandeis University for their help and suggestions on this work.

References

[1] J. Klausen, R. Libretti, B. W. K. Hung, and A. P. Jayasumana, "Radicalization trajectories: An evidence-based computational approach to dynamic risk assessment of "homegrown" jihadists," *Studies in Conflict & Terrorism*, pp. 1–28, 2018.

[2] R. Lara-cabrera, A. G. Pardo, K. Benouaret, N. Faci, D. Benslimane, and D. Camacho, "Measuring the radicalisation risk in social networks," *Special Section on Heterogeneous Crowdsourced Data Analytics*, vol. 5, pp. 10892–10900, 2017.

[3] C. D. S. E. Institute, "Insider threat best practices." https://resources.sei.cmu.edu/library/asset-view.cfm. Accessed on 28-June-2019.

[4] D. Edelman and M. Singer, "Competing on customer journeys," *Harvard Business Review*, November 2015.

[5] D. Edelman and M. Singer, "The new consumer decision journey." https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/the-new-consumer-decision-journey. Accessed on 28-June-2019.

[6] J. Jashinsky, S. Burton, C. Hanson, J. West, C. Giraud-Carrier, M. Barnes, and T. Argyle, "Tracking suicide risk factors through twitter in the us," *Crisis*, vol. 35, p. 51–59, 2014.

[7] R. Olson, "Suicide threats on social network sites" centre for suicide prevention." http://www.sprc.org/resources-programs/suicide-threats-social-networking-sites, 2011.

[8] J. Klausen, S. Campion, N. Needle, G. Nguyen, and R. Libretti, "Toward a behavioral model of "homegrown" radicalization trajectories," *Studies in Conflict & Terrorism*, vol. 39, no. 1, pp. 67–83, 2016.

[9] M. King and D. Taylor, "The radicalization of homegrown jihadists:a review of theoretical models and social psychological evidence," *Terrorism and Political Violence*, no. 23, pp. 602–622, 2011.

[10] "Facebook graph search." http://www.facebook.com/search/. Accessed on 28-June-2019.

[11] "Search twitters." https://developer.twitter.com/en/docs/tweets/search/overview. Accessed on 28-June-2019.

[12] "What is a graph database?." https://neo4j.com/developer/graph-database/. Accessed on 17-June-2019.

[13] Neo4j.com, "Neo4j graph platform." https://neo4j.com/. Accessed on 17-June-2019.

[14] M. Needham and A. Hodler, *Graph Algorithms*. No. 3, O'Reilly Media, Inc, 2019.

[15] Neo4j.com, "The neo4j graph algorithms user guide v3.5." https://neo4j.com/docs/graph-algorithms/current/. Accessed on 20-July-2019.

[16] M. Asiler and A. Yazici, "Bb-graph: A subgraph isomorphism algorithm for efficiently querying big graph databases," 2018.

[17] S. R. Muramudalige, *Investigative Pattern Detection in Large Heterogeneous Data (Tentative)*. PhD thesis, Colorado State University. In Progress.

[18] Neo4j.com, "Neo4j graph database sandbox." https://neo4j.com/sandbox-v2/. Accessed on 28-June-2019.

[19] "UK open data." https://data.gov.uk/. Accessed on 28-June-2019.

[20] B. Hung and A. Jayasumana, "Investigative simulation: Towards utilizing graph pattern matching for investigative search," *Proceedings of the Conference on Foundations of Open Source Intelligence and Security Informatics (FOSINT-SI)*, 2016.

[21] J. R. Ullmann, "An algorithm for subgraph isomorphism," *Journal of the Association for Computer Machinery*, vol. 23, pp. 31–42, 1976.

[22] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 10, pp. 1367–1372, 2004.

[23] V. Carletti, P. Foggia, A. Saggese, and M. Vento, "Challenging the time complexity of exact subgraph isomorphism for huge and dense graphs with vf3," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, p. 804–818, 2018.

[24] W. Fan, X. Wang, and Y. Wu, "Diversified top-k graph pattern matching," *Proceedings of the VLDB Endowment*, vol. 6, no. 13, pp. 1510–1521, 2013.

[25] S. Ma, Y. Cao, W. Fan, J. Huai, and T. Wo, "Strong simulation: Capturing topology in graph pattern matching," *ACM Transactions on Database Systems*, vol. 39, no. 1, 2014.

[26] B. Hung, A. P. Jayasumana, and V. W. Bandara, "Insight: A system to detect violent extremist radicalization trajectories in dynamic graphs," *Data Knowleage Engineering*, vol. 118, pp. 52–70, 2018.

[27] B. Hung, A. P. Jayasumana, and V. W. Bandara, "Finding emergent patterns of behaviors in dynamic heterogeneous social networks (accepted)," *IEEE Transactions on Computational Social Systems*, 2019.