GUDIE: a flexible, user-defined method to extract subgraphs of interest from large graphs

Maria Inês Silva, David Aparício, Beatriz Malveiro, João Tiago Ascensão, and Pedro Bizarro

Feedzai
{maria.silva, david.aparicio, beatriz.malveiro, joao.ascensao, pedro.bizarro}@feedzai.com

Abstract. Large, dense, small-world networks often emerge from social phenomena, including financial networks, social media, or epidemiology. As networks grow in importance, it is often necessary to partition them into meaningful units of analysis. In this work, we propose GUDIE, a message-passing algorithm that extracts relevant context around seed nodes based on user-defined criteria. We design GUDIE for rich, labeled graphs, and expansions consider node and edge attributes. Preliminary results indicate that GUDIE expands to insightful areas while avoiding unimportant connections. The resulting subgraphs contain the relevant context for a seed node and can accelerate and extend analysis capabilities in finance and other critical networks.

Keywords: graph expansions · message passing algorithms · banking networks · fraud detection · anti-money laundering

1 Introduction

Complex networks appear in many real-life systems, such as social, biological, or technological systems. Recurrent examples are small-world networks [1] characterized by small diameters and few links of separation between any two nodes, which leads to efficient exchanges at both local and global levels [2]. Banking networks, for example, are vast but have small diameters due to the existence of high-degree nodes, such as large merchants. These supernodes resemble superspreaders in epidemiology [3, 4, 5] or social media [6]. Even though being so pervasive, understanding these networks is difficult since they are typically large and exhibit complex collective behaviors.

When exploring such networks, it is common to use dedicated interfaces such as node-link diagrams [7]. Visualization platforms often follow the "Search, Show Context, Expand on Demand" paradigm [8] and require users to interact with the graph and expand nodes on-demand to uncover additional context. However, it is not always clear which expansions lead to the most relevant context.

Direct connections provide immediate context and are typically relevant. Some indirect connections may also be informative, albeit resulting in more complex subgraphs. In particular, expanding a few hops in large, small-diameter

networks can yield visually overwhelming graphs that contain mostly irrelevant connections.

To address the problem of generating relevant expansions, we propose GUDIE (Graph User-Defined Interest Expansions). This novel algorithm that extracts relevant context around a set of nodes in highly connected networks based on user-defined criteria. We devise GUDIE to assist financial crime investigations in banking networks. Nonetheless, its reliance upon user-defined criteria makes it adaptable to different datasets and use-cases. GUDIE is a message-passing algorithm, which makes it parallel by design and scalable to large networks.

The remaining of this paper is structured as follows. Section 2 details GUDIE. Section 3 contains preliminary results. We review related work in Section 4. Finally, Section 5 summarizes conclusions and future directions.

2 Method

GUDIE[1] is a message-passing algorithm that returns the most interesting expansion for each node in a list of nodes, based on user-defined interest. In this section, we discuss user-defined interest (Section 2.1), desirable properties (Section 2.2), and provide an overview of our method (Section 2.3).

2.1 User-Defined Interest

Interest is data- and use-case specific. Hence, considering user-defined criteria for the expansions is an integral part of GUDIE, and it ensures flexibility and adaptability to different networks.

GUDIE's expansions rely upon user-defined interest functions and the structural properties of the graph. The user provides two types of interest functions: node or **v**ertex interest (**V**UDIE[2]) and edge or **l**ink interest (**L**UDIE[3]). Both interest functions receive information about the graph and provide a score between 0 and 1. Higher scores stand for higher interest.

As an example, let us consider banking networks. Relevant *node* information may include node type (e.g., card, device, IP), labels (e.g., past alerts or suspicious activity), recent activity (e.g., active or dormant entities), and topological features (e.g., degree, clustering coefficient). Past labels can also influence *edge* interest (e.g., fraud labels) alongside edge weight (e.g., transaction amount) and recency. Finally, edge interest can depend on its relationship with nodes, as similarity with other edges connected to the same entities.

2.2 Desirable Properties

We design GUDIE to satisfy specific properties. Mainly, we aim to attribute higher interest to (P1) nodes that are closer to the seed, (P2) nodes in high-interest

[1] Acronym for **G**raph **U**ser-**D**efined **I**nterest **E**xpansions.
[2] Acronym for **V**ertex **U**ser-**D**efined **I**nterest **E**xpansions.
[3] Acronym for **L**ink **U**ser-**D**efined **I**nterest **E**xpansions.

areas, and (P3) nodes connecting the seed to high-interest areas. Moreover, we want to ensure that the number of connections alone does not automatically drive interest. Thus (P4) node interest does not necessarily increase with the node degree. Alternatively, the user-defined node interest function can consider degree centrality when desirable.

2.3 Method Overview

GUDIE (Algorithm 1) consists of four main steps.[4]

Algorithm 1 GUDIE

Input: Graph G; list of seed nodes $S \subseteq V(G)$; node interest function VUDIE : $V(G) \to [0, 1]$; edge interest function LUDIE : $E(G) \to [0, 1]$; number of interest propagation hops h;interest propagation function ϕ; interest aggregation function γ; decay function θ; interest threshold $k \in [0, 1]$.
Output: GraphUnits $\mathcal{S}$ (list of subgraphs), one for each seed.

1: $I_V, I_E \leftarrow$ Initialize$(G, \text{VUDIE}, \text{LUDIE})$
2: $I_V^h \leftarrow$ InterestPropagation$(G, I_V, I_E, h, \phi, \gamma)$
3: $\mathcal{G} \leftarrow$ SeedsExpansion(G, I_V^h, S, θ, k)
4: $\mathcal{S} \leftarrow$ ObtainGraphUnits$(\mathcal{G})$

Initialization. Node and edge interest scores, I_V and I_E, respectively, are computed according to the interest functions VUDIE and LUDIE. At this point, every node and edge in G has an interest score.

Interest Propagation. GUDIE propagates the interest through the network. Here, nodes message their neighbors their interest score. The interest propagation process runs for h hops and considers node interest, I_V, and edge interest, I_E. According to the interest propagation function, nodes split their node interest among their neighbors, ϕ, and update their interest according to the interest aggregation function, γ. After this step, we have a new propagated interest score I_V^h.

Seed Expansion. GUDIE runs the expansions for each seed $s \in S$. The seed expansion process runs on top of G and the propagated node interest I_V^h. The expansion starts by computing the minimum allowed interest for the seed using the interest expansion tolerance, k. During expansion, distant nodes from the seed are penalized according to the distance decay function θ. At the end of the seed expansion process, each node contains the expansions traversing them. We store this information in $\mathcal{G}$.

[4] Appendix A further details the GUDIE method, including its inputs, design choices and implementation.

Obtain GraphUnits. GUDIE uses a map-reduce operator to obtain the subgraphs.

3 Preliminary Results

3.1 Prototypical Examples

This section showcases five examples that mimic known patterns in banking networks. We analyze the expansions obtained with GUDIE.

We consider banking networks with four entities types corresponding to four different node types: customers, merchants, devices, and IP addresses. Two nodes are connected if there is at least one shared financial transaction. Edges have three attributes: the transaction label (i.e., fraudulent or legitimate), the transaction timestamp, and the transaction amount.

In all cases, we use customer *C1* as the seed node. We use a constant interest function for nodes, $I(N_i) = 1.0$, and an edge interest function that depends on the fraud rate and the time-weighted amount of the transactions.[5]

Uninteresting edges Consider two transactions made by customer *C1*: one legitimate and one high amount fraudulent payment. Considering how the interest functions were defined, we expect GUDIE to expand *only* to the entities involved in the high-amount fraudulent transaction, as illustrated in Figure 1.

Customer	Merchant	Device	IP	Amount	Fraud?
C1	M1	D1	IP1	$1	No
C1	M2	D2	IP2	$100M	Yes

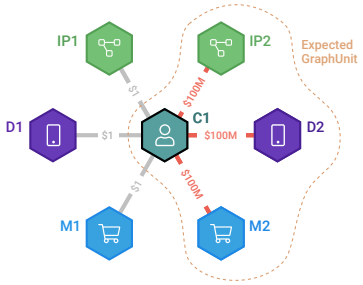

Fig. 1: Ignoring uninteresting edges (Example 1).

In the initialization step, the interest given to the legitimate and low-amount edges is low, while the interest given to the fraudulent and high-amount edges is high (Figure 2). GUDIE propagates the interest and, when the expansion is complete, the GraphUnit contains only the customer *C1* and the entities involved in the high-amount fraudulent transaction.

[5] Refer to Appendix B for full configurations.

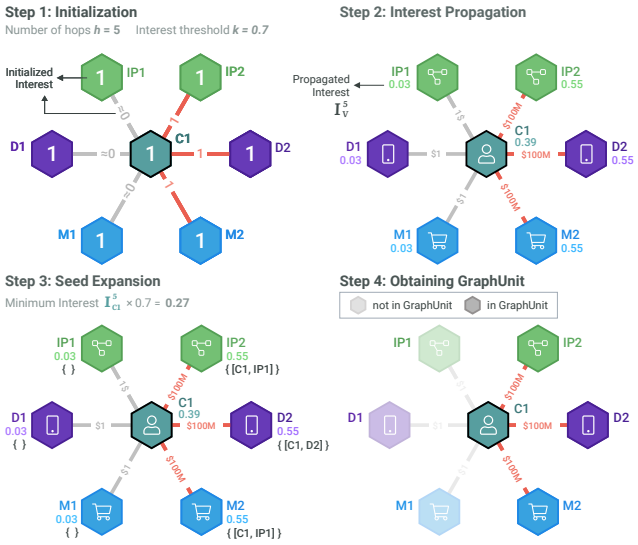

Fig. 2: Steps to obtain the GraphUnit for Example 1.

Interesting indirect connections The second example covers a case where the customer *C1* makes a legitimate purchase on merchant *M1*. However, fraudulent transactions were previously made by customer *C2* on this very same merchant, as illustrated in Figure 3. This example tests whether GUDIE is capable of connecting *C1* to fraudulent activity beyond its direct connections.

As GUDIE attributes a high-interest score to the connected merchant, the expansion reaches the fraudulent subgraph. The resulting GraphUnit matches our expectations and shows that GUDIE is capable of expanding to interesting indirect neighbors.

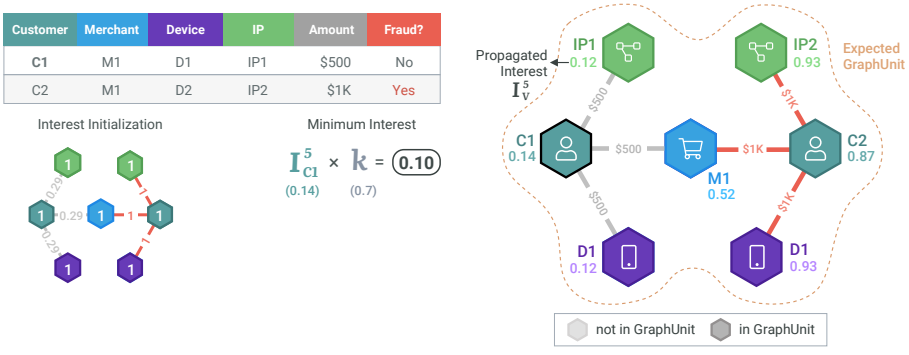

Fig. 3: Interesting indirect nodes (Example 2).

Irrelevant supernodes The third example consists of a customer *C1* that made a legitimate purchase on a large merchant with a low fraud rate of 10%, as illustrated in Figure 4. Due to the size of the merchant and its low fraud rate, transactions made by other customers are not a helpful context for investigators.

As expected, GUDIE attributes a low-interest score to the merchant. Hence, the expansion does not pursue this path: the final GraphUnit excludes the connections of the merchant, including the one associated with a fraudulent transaction. This example demonstrates that GUDIE does not show connections to uninteresting high-degree nodes.

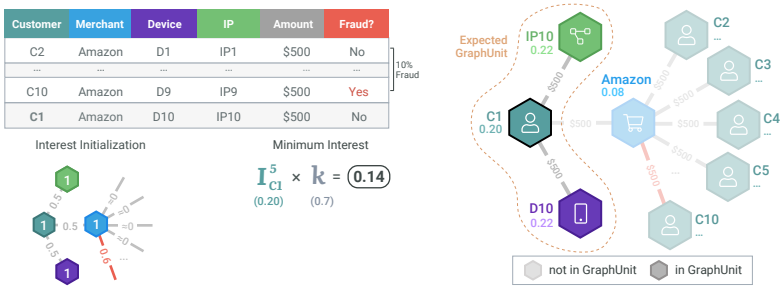

Fig. 4: Irrelevant supernodes (Example 3).

Relevant supernodes Let us consider a variation of the previous case with a fraud rate of 40%. Given this higher fraud rate, we expect GUDIE to include the merchant in the GraphUnit.

Accordingly, we see in Figure 5 that GUDIE attributes a high-interest score to the fraudulent merchant and expands to it. However, the interest is not large enough to continue the expansion to other customers connected to the merchant. Thus, GUDIE can expand to supernodes of high interest.

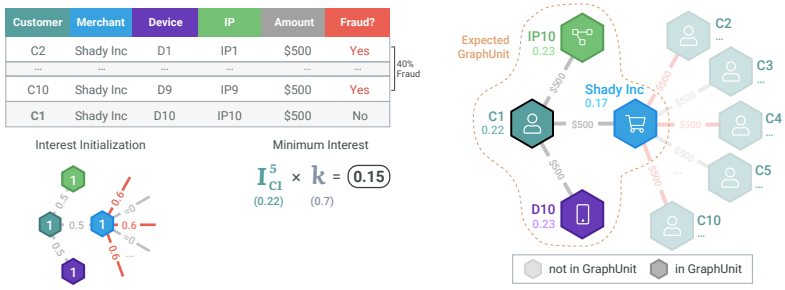

Fig. 5: Relevant supernodes (Example 4).

Interesting areas through uninteresting edges Finally, we test whether GUDIE can reach high-interest areas connected to the seed node by low-interest nodes. Figure 6 illustrates this scenario where none of *C1*'s direct connections are interesting, as they represent low amount legitimate purchases. However, the merchant connected to *C1* is connected to a fraudulent customer *C2*.

As expected, GUDIE increases the merchant's interest and reduces the interest of the other direct connections. Accordingly, the expansion reaches the fraudulent customer *C2*, allowing GUDIE to expand to a distant region of high interest.

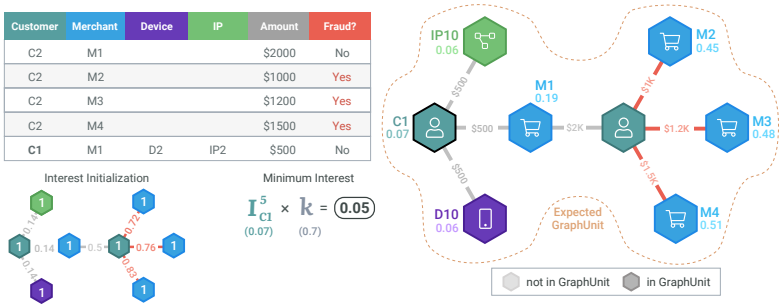

Fig. 6: Interesting areas through uninteresting edges (Example 5).

3.2 Usability

This section provides a usability analysis of GUDIE. Although inspired by Feedzai's Visual Insights platform, *Genome*, we believe it applies to other domains where human investigators analyze large networks, such as financial networks, social media, or epidemics, for example.

In financial investigations, analysts use dedicated interfaces[6] to review alerted cases. Usually, in less than five minutes, analysts must review the data, reason about potential fraud scenarios, and decide whether the case is suspicious. Interactive graph diagrams provide a quick visual overview of the case by immediately showing relevant connections of its entities.

However, without GUDIE, the initial graph presented to the analyst consists of only the entities present in the case. This information is often insufficient to make a decision. In order to uncover the relevant context, the analyst performs the following tasks: (T1) expand[7] interesting nodes, (T2) remove irrelevant new nodes, (T3) repeat T1-T2 until there is enough context.

This manual trial-and-error exploratory analysis is time-consuming and far from trivial, potentially compromising the target review time. GUDIE removes the need to perform tasks (T1–T3) by automatically showing the relevant *GraphUnit*. Consequently, we expect GUDIE to enable faster decision-making.

Furthermore, we envision GUDIE assisting complex linked analysis by enabling more concise expansions. Expanding an interesting node (T1) could yield

[6] Refer to Appendix B.2 for an illustrative example of this user interface.
[7] An expansion operation adds connections (nodes and edges) to the view.

its *GraphUnits*, encapsulating its relevant context while automatically hiding uninteresting connections (T2).

4 Related work

Techniques to deal with large networks include sampling [9], network partitioning – namely local clustering [10] and community detection [11], and studying diffusion and influence [12]. GUDIE aims to extracts the relevant context around a node. We believe our method is especially suited for large, dense, small-word networks. Evidence on many self-organizing social systems suggests densification and shrinking network diameters over time [13].

Local graph clustering shares similarities to our problem since its goal is to identify a cluster near a given node [10]. A local cluster is a group of nodes around a seed node with high connectivity between local nodes and low connectivity to nodes outside the cluster.

Connectivity is commonly defined in terms of edge conductance [14, 15]. Recent work has expanded the concept of connectivity to consider high-order structures, which relate more closely to our work. Motif conductance typically encodes high-order structures by measuring the conservation of specific subgraph patterns, such as triangles, cycles, or stars, inside the cluster [16, 17, 10].

Even though motif conductance is more flexible than edge conductance, current work concerns a single subgraph. Thus it does not work when it is inadequate to represent interest as a single subgraph. On the other hand, GUDIE is more flexible since it allows for user-defined interest functions.

Additionally, because these methods optimize motif conductance only, they do not penalize nodes distant from the seed. GUDIE takes into account distance through a decay function.

5 Conclusion

We propose GUDIE to extract node context from large, highly connected networks. To the best of our knowledge, GUDIE is the first method that employs user-defined criteria to retrieve the relevant context around a seed node.

We present preliminary results. We include five examples to showcase desirable properties. In all examples, GUDIE extracts the desired subgraph. We also put forward usability considerations. We designed GUDIE to be highly adaptable to other datasets and use-cases through user-defined interest. Although the experiments reflect banking networks and financial crime investigations, we believe the insights apply to other network types.

Shortly we plan to test GUDIE on real-world banking networks. Mainly, we aim to evaluate GUDIE's scalability and conduct user tests to assess its impact on the speed and accuracy of financial crime investigations.

Moreover, we believe there is an opportunity to test GUDIE on other domains, particularly in the context of other large, dense, small-world networks.

Bibliography

[1] Stanley Milgram. The small world problem. *Psychology today*, 2(1):60–67, 1967.

[2] Vito Latora and Massimo Marchiori. Efficient behavior of small-world networks. *Phys. Rev. Lett.*, 87:198701, Oct 2001.

[3] Richard A Stein. Super-spreaders in infectious diseases. *International Journal of Infectious Diseases*, 15(8):e510–e513, 2011.

[4] Sara H Paull, Sejin Song, Katherine M McClure, Loren C Sackett, A Marm Kilpatrick, and Pieter TJ Johnson. From superspreaders to disease hotspots: linking transmission across hosts and space. *Frontiers in Ecology and the Environment*, 10(2):75–82, 2012.

[5] Serina Chang, Emma Pierson, Pang Wei Koh, Jaline Gerardin, Beth Redbird, David Grusky, and Jure Leskovec. Mobility network models of covid-19 explain inequities and inform reopening. *Nature*, pages 1–6, 2020.

[6] Sen Pei, Lev Muchnik, José S Andrade Jr, Zhiming Zheng, and Hernán A Makse. Searching for superspreaders of information in real-world social media. *Scientific reports*, 4(1):1–12, 2014.

[7] René Keller, Claudia M. Eckert, and P. John Clarkson. Matrices or node-link diagrams: Which visual representation is better for visualising connectivity models? *Information Visualization*, 5(1):62–76, 2006.

[8] F. van Ham and A. Perer. "Search, Show Context, Expand on Demand": Supporting Large Graph Exploration with Degree-of-Interest. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):953–960, 2009.

[9] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636, 2006.

[10] Dongqi Fu, Dawei Zhou, and Jingrui He. Local motif clustering on time-evolving graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 390–400, New York, NY, USA, 2020. Association for Computing Machinery.

[11] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[12] Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4):1–37, 2012.

[13] Jurij Leskovec. *Dynamics of large networks*. PhD thesis, Carnegie Mellon University, School of Computer Science, Machine Learning . . . , 2008.

[14] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486, 2006.

[15] Daniel A. Spielman and Shang-Hua Teng. A Local Clustering Algorithm for Massive Graphs and Its Application to Nearly Linear Time Graph Parti-

tioning. *SIAM Journal on Computing*, 42(1):1–26, January 2013. Publisher: Society for Industrial and Applied Mathematics.

[16] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 555–564, New York, NY, USA, 2017. Association for Computing Machinery.

[17] D. Zhou, J. He, H. Davulcu, and R. Maciejewski. Motif-preserving dynamic local graph cut. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1156–1161, 2018.

A Supplemental Material – GUDIE

A.1 User Inputs

Below, we detail GUDIE's user parameters, besides the interest functions, VUDIE and LUDIE (defined in Section 2).

Interest propagation hops (h) This parameter dictates how many hops away two nodes can be in order for them to influence each other's interest. In essence, we want to enable the user to control how long-reaching the interest propagation should be. For instance, in a connected graph, $h = diameter(G)$ guarantees that all nodes influence each other's interest at least a small amount proportional to their distance.

Interest threshold (k) Controls the seed expansion process (i.e., when to stop). The threshold is expressed as a relative value of the seed's original interest. For instance, if $k = 0.8$, GUDIE will ignore expansions that are 20% less interesting than the seed's initial interest. In the extremes, if $k = 1$, expansions can only go to nodes at least as interest as the seed, s, and if $k = 0$ all expansions are allowed.

Interest aggregation function (γ) This function specifies how a node combines its interest, obtained from the previous iteration, with the interest received from its neighbors in the current iteration.

A possibility for γ follows in Equation 1:

$$\gamma_1(I_n^{h-1}, M_n) = \frac{I_n^{h-1}}{2} + \frac{\sum_{i_{n,m} \in M_n} i_{n,m}}{2|M_n|} \tag{1}$$

γ_1 gives equal weight to the previous node interest, I_n^{h-1}, and to the average of the messages received in the current iteration, M_n.

It is important to note that since γ_1 is first aggregating the messages received, it does not add more interest to nodes just more having more connections. Recall that this was one of the desired properties of the method.

Other functions might give different weight to I_n^{h-1} and use different aggregations of the messages M_n, such as the *max* or *min*, for example. However, if one were to use a *sum* aggregation instead of a *mean* aggregation, super-nodes would receive more interest just for having more neighbors.

Decay function (θ) Specifies the weight decay of a node's interest relative to its distance to the seed.

Equations 2 and 3 introduce two possibilities that progressively decrease the interest of more distant nodes follow:

$$\theta_1(L) = 1 - |L|^{-1} \tag{2}$$

Equation 3 decreases faster than Equation 2, which might be more or less adequate depending on the data, the use-case, and the desired expansions (more or less local).

$$\theta_2(L) = 1 - e^{1-|L|} \tag{3}$$

A.2 Other Parameters

Interest propagation function (ϕ) Equation 4 contains the interest propagation function. As it is defined, ϕ has two properties that make it well suited for interest propagation.

$$\phi(n, m) = I_n \times I_{n,m} \tag{4}$$

Because the function is monotonic increasing on the nodes interest I_n, nodes close to high-interest areas receive messages with high interest values from their neighbors and, therefore, have high propagated interests.

Moreover, because the function is monotonic increasing on the edge interest score $I_{n,m}$, messages received via low-interest interest edges are penalized and vice-versa. Thus, a node connected to a high-interest area via a low-interest edge will have a lower propagated interest than a node connected to the same area via an high-interest node.

A.3 Initialization

Algorithm 2 describes GUDIE's initialization. GUDIE uses VUDIE (lines 2–3) and LUDIE (lines 3-4) to assign an interest to each node and edge in G, respectively. At the end of the process, GUDIE outputs the node interest for all nodes in G, I_V, and the edge interest for all edges in G, I_E. The values can be appended to the graph to build a new weighted graph or used as a separate data-structure (e.g., a dictionary mapping nodes and edges to their respective interest).

Algorithm 2 GUDIE initialization.

Input: Graph G; node interest function VUDIE : $V(G) \rightarrow [0, 1]$; edge interest function LUDIE : $E(G) \rightarrow [0, 1]$.
Output: Nodes' interest I_V and edges' interest I_E.

```
1: function INITIALIZE(G, VUDIE, LUDIE)
2:     for all n ∈ V(G) do
3:         I_n = VUDIE(n)
4:     for all (n, m) ∈ E(G) do
5:         I_{n,m} = LUDIE(n, m)
6:     return I_V, I_E
```

A.4 Interest propagation

The interest propagation step is detailed in Algorithm 3. Interest propagation takes as input the graph G, alongside initial node and edge interest, I_V and I_E, h, ϕ, and γ, and computes the propagated interest, I_V^h.

Algorithm 3 GUDIE interest propagation.

Input: Graph G; nodes' interest I_V; edges' interest I_E; interest propagation hops h; interest aggregation function γ.
Output: Updated nodes interest I_V^h (after interest propagation).

```
1: function INTERESTPROP(G, I_V, I_E, h, φ, γ)
2:     I_V^0 ← I_V
3:     for h hops do
4:         for all n ∈ V(G) do
5:             M_n ← ∅
6:         for all (n, m) ∈ E(G) do
7:             i_{n,m} ← φ(n, m)
8:             M_n· APPEND(i_{n,m})
9:         for all n ∈ V(G) do
10:            I_n^h ← γ(I_n^{h−1}, M_n)
11:    return I_V^h
```

First, GUDIE sets the initial interest I_V^0 as the nodes' initial interest, I_V (line 2). Then, there are h iterations of interest propagation (line 3–8). At each iteration, we initialize empty each node's set of received messages, M_n (lines 4–5). Then, all nodes send messages to their neighbors through their edges (lines 6–8). Messages contains the interest $i_{n,m}$, obtained using the interest propagation function, ϕ (line 7).

After a node m receives a message from neighbor n, it updates its pool of received messages to contain $i_{n,m}$ (line 8). After all messages for iteration h

have been exchanged, each node updates its node interest using the aggregation function γ (lines 9–10).

A.5 Seed expansion

The seed expansion step takes as input the graph G, the propagated node interest I_V^h, the list of seed nodes S, and parameters θ and k to obtain the list of expansions, $\mathcal{G}$, traversing each node $n \in V(G)$ starting from one of the seeds $s \in S$. Algorithm 4 describes the seed expansion process.

Algorithm 4 GUDIE seed expansion.

Input: Graph G; nodes interest I_V^h; list of seed nodes $S \subseteq V(G)$; decay function θ; interest threshold $k \in [0, 1]$.
Output: Expansions $\mathcal{G}$ traversing each node $n \in V(G)$.

```
1: function SeedsExpansion(G, I_V^h, S, θ, k)
2:     U ← ∅
3:     for all n ∈ V(G) do
4:         G_n ← ∅
5:     for all s ∈ S do
6:         δ ← I_s^h × k
7:         P ← [s]
8:         g_s ← Expansion(δ, P)
9:         G_s ← {g_s}
10:        U ← U ∪ {s}
11:    while U is not empty do
12:        U' ← ∅
13:        for all n ∈ U do
14:            for all (n, m) ∈ E(G) do
15:                for all g_n ∈ G_n do
16:                    if m ∉ g_n. path and
17:                        (1 − θ(g_n· path)) × I_m^h ≥ g_n· minInterest then
18:                        g_m ←Expansion(g_n· minInterest, g_n· path ∪{m})
19:                        if g_m ∉ G_M then
20:                            G_m ← G_m ∪ {g_m}
21:                            U' ← U' ∪ {m}
22:        U ← U'
23:    return G
```

Initialization The first step is to initialize the relevant variables.

The set of updated nodes, U, and the expansions traversing each node, $\mathcal{G}_n$, are set to empty (lines 1–4).

The minimum interest, δ, is derived from the seed interest, I_s^h, and the tolerance, k (line 6). The path, P, is initialized with the single seed, s (line 7). We create an expansion, g_s, from δ and P (line 8) and add it to the list of all expansions going through s, $\mathcal{G}_f$ (line 9).

Finally, we add s to the set of updated nodes U (line 10).

Expansion Iterative expansions run until no nodes are updated with new expansions in the previous iteration (lines 11-22).

In each iteration, all nodes that received updates in the previous iteration check if there are valid extensions going through their neighbors (lines 13-21).

If the conditions are satisfied, node m creates a new expansion, g_m, containing the minimum interest allowed by the seed, (gn.MININTEREST), and updates the path traversed, (g_n.PATH), augmented by $\{m\}$.

If the expansion g_m was already found in previous iterations, it is discarded (line 19). If it is new, g_m is added to the list of traversed paths, $\mathcal{G}_m$ (line 20), and the node m is added to the list of nodes that have been updated in the current iteration, U' (line 21).

At the end of each iteration, the list of nodes to be explored is updated (line 22) and, if the list is not empty, the process continues (lines 11-22).

A.6 Obtain GraphUnits

Obtaining GraphUnits is done using a map-reduce operation, described in Algorithm 5 and Figure 7.

Algorithm 5 GUDIE map-reduce operation to obtain GraphUnits.

The mapper emits an intermediate seed-expansion pair for each node in the graph. The reducer obtains the GraphUnit for each seed.

```
 1: function ObtainGraphUnits(𝒢)
 2:     procedure map(𝒢_n)
 3:         for all g_n ∈ 𝒢_n do
 4:             L ← g_n· path
 5:             s ← L[0]
 6:             emit(seed s, expansion L)
 7:     procedure reduce(s, ℒ = [L_1, L2, ...])
 8:         S = ∅
 9:         for all L ∈ ℒ do
10:             S ← S ∪ L
11:         emit(s, S)
```

The mappers traverse their corresponding list of expansions, $\mathcal{G}_n$ (lines 2-6), and, for each one of them, g_n (line 3), obtain their path, L (line 4), seed, s (line 5), and emit the seed-expansion pair (line 6).

Thus, at the end of the mapping process, all pairs of seed-expansion have been produced.

Finally, for each seed s, the reduce operator combines the resulting paths and obtains the GraphUnit.

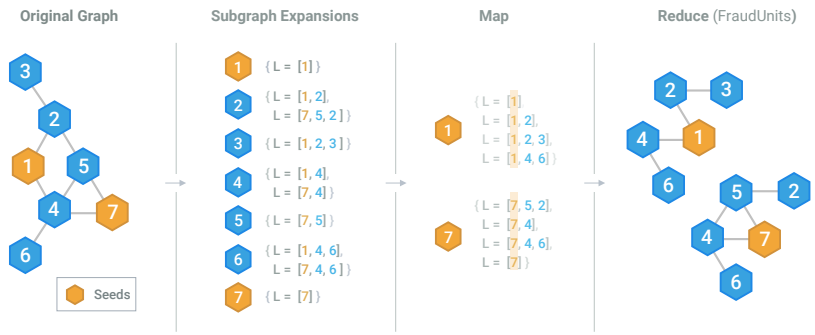

Fig. 7: Map-reduce operator to obtain GraphUnits.

B Supplemental Material – Preliminary Experiments

We initialize GUDIE with the following parameters:

- Number of hops, $h = 5$
- Interest threshold, $k = 0.7$
- Aggregation function, γ: Equation 1
- Decay function, θ: Equation 3

B.1 Edge Interest (LUDIE)

Let us define Ω_t as the most recent timestamp in the dataset (in seconds) and $E_{i,j}$ as an edge representing transactions between nodes i and j.

For each transaction $e_{i,j} \in E_{i,j}$, we define its weighted amount as

$$w\_amount(e_{i,j}) = amount(e_{i,j}) * e^{-\Delta t}$$

where

$$\Delta t = (\Omega_t - timestamp(e_{i,j}))/one\_week\_in\_seconds$$

From here, we define the time-weighted amount of $E_{i,j}$ as follows

$$tw\_amount(E_{i,j}) = \sum_{e_{i,j}} w\_amount(e_{i,j})$$

Additionally, for each $E_{i,j}$, let $fraud\_rate(E_{i,j})$ be the ratio of fraudulent transactions between nodes i and j. Assuming that Ω_t^{max} is the maximum time-weighted amount in the graph, we define the edge interest function as

$$I(E_{i,j}) = \frac{tw\_amount(E_{i,j})}{2\Omega_t^{max}} + \frac{fraud\_rate(E_{i,j})}{2}$$

B.2 Usability

Consider the following interface to review suspicious financial crime cases illustrated in figure 8. The interface comprises a single page that displays the details of the alerted case and associated transactions, alongside customer information and the interactive graph diagram. This tool is optimized so that the analyst can review the case accurately and fast.

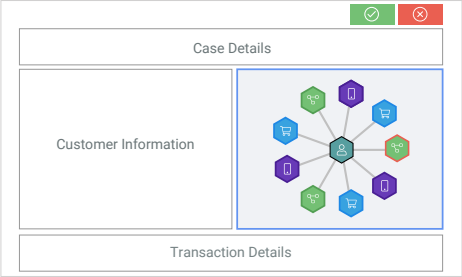

Fig. 8: Illustration of graph placement in the interface used by fraud analysts to review alerted cases.