Discovering Fraudulent Card Transactions With Higher Order Graph Embeddings Over Neo4j

Christos Hadjisofokelous[1], Georgios Drakopoulos[2(✉)], Spyros Sioutas[1], and Phivos Mylonas[2]

[1] Computer Engineering and Informatics Department, University of Patras, Patras, Greece
{chatzisofokleous,sioutas}@ceid.upatras.gr
[2] Informatics and Computer Engineering Department, University of West Attica, Aigaleo, Greece
{georgedrakopoulos,mylonasf}@uniwa.gr

Abstract. Credit card transactions, especially when linked to smart devices and the IoT ecosystem in general, are one of the drivers of contemporary digital economy as well as a major indicator of the overall financial activity. As such as well as for a plethora of other reasons it is imperative that fraudulent transactions be efficiently and reliably discovered. Because of their interconnected and time-dependent nature, a graphic representation not only is convenient, but also lends itself to machine learning strategies. To this end one viable approach is to construct a framework consisting of three steps. First, at each vertex a vector containing first and higher order attributes is embedded, then vertices are clustered, and finally vertex classification is done. As a concrete example three graph partitioning algorithms were selected, namely kNN, DBSCAN, and spectral clustering, whereas vertex clustering has been performed through logistic regression. The experimental results corroborate the efficiency of the abovementioned framework and are encouraging for the development of more higher order fraudulent transaction methods towards a more robust and highly reliable digital economy.

Keywords: Graph clustering · DBSCAN · kNN · Higher order data · Logistic regression · Graph machine learning · Graph queries · Graph databases · Neo4j · Cypher · Fraudulent transactions · Digital economy

1 Introduction

With the worldwide growth of digital economy credit card and electronic payments are becoming increasingly importance for a number of reasons including speed, ease of access, and traceability. However, this also creates new attack vectors such as identity theft and fraud. To counter the latter, systems mining the vast ocean of credit card transaction data have already been developed and deployed. The analytics of the latter depend heavily on the representation of the

Published by Springer Nature Switzerland AG 2025
I. Maglogiannis et al. (Eds.): AIAI 2025, IFIP AICT 755, pp. 424–436, 2025.
https://doi.org/10.1007/978-3-031-96239-4_31

transaction data as this dictates the nature and complexity of the subsequent processing. Moreover, because of the inherently flexible and distributed nature of the information stored in graphs, a plethora of machine learning (ML) strategies can be applied, each tailored to the semantics of the underlying domain.

The principal motivation behind this work is the growth of the significance of electronic payments in digital economy as stated earlier and consequently their safety is paramount for the robustness thereof. Given that in conjunction with the smart devices and the IoT ecosystem such payments offer a convenient and safe alternative to physical currency, it becomes evident that sagefuarding electronic payments is paramount to a robust and reliable digital economy.

The primary research contribution of this conference paper is a modular graph theoretic framework for discovering fraudulent credit card transactions with a linear architecture consisting of the following three steps.

- At each vertex is created a vector containing first and higher order attributes pertaining to both the structure and the function of the transaction graph.
- Then vertices are clustered in a space of a higher order dimension determined by the number of attributes to isolate various card transaction types.
- Finally a classifier operates on the clusters to determine which ones are most likely to contain fraudulent transactions.

The remainder of this work is structured as follows. In Sect. 2 the recent scientific literature regarding graph mining and graph signal processing (GSP) are briefly overviewed. In Sect. 3 the proposed graph processing framework for discovering fraudulent transactions is described, while in Sect. 4 the results obtained from it are presented. Possible future research directions are explored in 5. Technical acronyms are explained the first time they are encountered in the text. Also the terms *attribute* and *feature* are used interchangeably throughout the text. Finally, the notation of this work is summarized in Table 1.

Table 1. Notation synopsis.

Symbol	Meaning	First in
$\triangleq$	Definition or equality by definition	Eq. (5)
$\Gamma(\cdot)$	Vertex neighborhood	Eq. (4)
$\{s_1, \ldots, s_n\}$	Set with elements $s_1, \ldots, s_n$	Eq. (5)
$\lvert\cdot\rvert$	Set cardinality functional	Sect. 3
$\lVert\cdot\rVert$	Matrix or vector norm	Eq. (12)
$\mathbf{I}$	Identity matrix	Eq. (9)
$\text{logit}(\cdot)$	Binary logit function	Eq. (14)

2 Previous Work

Graph mining is a vast research field [18,19,22] which currently focuses on diverse topics such as power laws describing graphs [38], efficient subgraph mining [27],

community discovery through ML [32] and autoencoders [36], Boolean semiring factorization [14], and regularized nonnegative matrix factorization [3], adjacency matrix factorization [25], graph approximation through polar factorization [11], link prediction [21], vertex classification [37], and graph clustering [34] as well as multiview clustering [26]. Applications include analysis of academic networks [39], ontology matching [29], clustering trajectories in social graphs [15], digital health [24], drug discovery [40], and botnet discovery [23]. Clustering applications include partitioning the base of a cultural game [10], using multiple similarity metrics encoded as tensors [17], partitioning MBTI personalities with self organizing maps (SOMs) [16], and power iteration graph clustering [9]. Graph neural networks (GNNs) [31,35] implemented in PyTorch [1,20] are currently the prime ML methodology applied on graphs [28,42]. Neo4j [2] is a graph database which has been applied to problems such as processing PubMed documents [13], cyber security analysis [4], and data science [5].

GSP is a cross-disciplinary field [43] encompassing topics such as graph spectral wavelets [33], graph Fourier transform [41], graph Kalman filter [6], gradient graph Laplacian [7], and variational Bayesian estimation [8]. Applications include brain science [30] and mining Industry 4.0 process graphs [12].

3 Framework

3.1 Attributes

Graphs are excellent vehicles for embedding attributes in their vertices or edges. In the proposed approach not only there were attributes in the vertices, but also the types of the vertices and the edges were an important part of the design conveying information themselves. Specifically, a transaction graph was created in Neo4j where each vertex was a person, a credit card, a merchant, or a transaction and each edge respresented the following relations between them:

- **[:HAS_BOUGHT_AT]:** This type of edge indicates that a specific person and a specific merchant have made a transaction.
- **[:OWNS_CARD]:** This type of edges denotes the owner of a specific credit card as a person can be linked to multiple cards.
- **[:USED_IN]:** This type of edges indicates that a specific card has been used in a given transaction. For each transaction there was only one card.

As it can be seen, not every type of vertex is connected to every other one. This not only allows for a sparse graph reducing memory requirements, but also higher granularity as many elements have their own representation and decluttered visualization at the expense of longer Cypher queries. For instance, once transactions have been flagged as suspicious or not, the query of Fig. 1 lists the ten merchants who have the highest ratio of possibly fraudulent transactions.

Depending on its nature, a set of first and higer order attributes were placed in each vertex. The first order attributes pertain to the vertex itself, are primarily functional, and do not depend on neighboring or other ones. These features are listed in Table 2. Moreover, in Fig. 2 the Cypher code for creating the embeddings

```
MATCH (m:Merchant)<-[t:HAS_BOUGHT_AT]-(p:Person)
WITH m.name AS Merchant,
     COUNT(t) AS TotalTransactions,
     SUM(CASE WHEN t.is_fraud = 1 THEN 1 ELSE 0 END) AS FraudulentTransactions
RETURN Merchant,
       FraudulentTransactions,
       TotalTransactions,
       toFloat(FraudulentTransactions) / TotalTransactions AS FraudRatio
ORDER BY FraudRatio DESC
LIMIT 10
```

Fig. 1. Cypher query for the merchants with the highest suspicous transaction ratio.

Table 2. Embedded first order vertex attributes.

Attribute	Meaning
transactionAmount	Amount of the transaction
timeStap	Date and time of the transaction
merchantCategoryCode	Type of business
creditCard	Information about the card such as issuer
transactionFrequency	How often a particular card is used
geoLocation	The locations associated with the card and the merchant
name	Card holder name

Table 3. Embedded higher order vertex attributes.

Attribute	Meaning
walkLength	The length of a random walk starting from that vertex
walksPerNode	Number of random walks starting from that vertex
inOutFactor	Balance between following inbound and outbound edges
returnFactor	Probability that random walks return to starting vertex

of the higher order attributes is shown. The latter are directly related to the graph structure such as the number of paths crossing a given vertex and they are explained in Table 3.

```
CALL gds.node2vec.write('myFraudDetectionGraph', {
  writeProperty: 'node2vecEmbedding',
  embeddingDimension: 64, //can be adjusted
  walkLength: 10, //can be adjusted
  walksPerNode: 10, //can be adjusted
  inOutFactor: 1.0, //can be adjusted
  returnFactor: 1.0 //can be adjusted
})
YIELD nodePropertiesWritten
```

Fig. 2. Code for the higher order attributes.

3.2 Graph Clustering

Three graph clustering algorithms have been used in the context of this conference paper. The ubiquitous kNN scheme shown in Algorithm 1 and DBSCAN described in Algorithm 2, while the spectral clustering as well as some possible variations are explained in Eqs. (3) and (4) and in the subsequent analysis.

The kNN algorithm is simple in its design and concept. Specifically, it assigns iteratively to each unlabeled data point the label of that of the majority of its k closest labeled neighbors according to the rule of Eq. (1). Said neighbors are determined by a given proximity metric depending on the underlying domain.

$$\ell_i \;=\; \mathrm{maj}\left[\{\ell_{j_1}, \ldots, \ell_{j_k}\}\right] \tag{1}$$

Algorithm 1. kNN clustering algorithm.

Require: Distance metric g; Termination criterion τ; Parameter k
Ensure: Unlabeled data points are labeled
  repeat
    for all unlabeled data points **do**
      find the k closest neighbors based on g
      assign ℓ_i based on (1)
    end for
  until τ is **true**
  return

DBSCAN relies heavily on the density, expressed as the number of points in a region of the data space, as well as on the reachability between important points termed *core points* as deemed by their local density. Specifically, if the number of points n_i around a given point $\mathbf{x}_i$ at a radius of ε is higher than n for a distance metric h, then x_i is deemed as a core point. Then, if a sequence of neighboring core points exists to a data point, then it is assigned to that cluster,

otherwise it is considered as noise. Thus DBSCAN results in a partitioning based on the transitivity of reachability between core and non-core points.

$$n_i \triangleq |\{\mathbf{s} \mid h(\mathbf{s}, \mathbf{x}) \leq \varepsilon\}| \tag{2}$$

Algorithm 2. DBSCAN clustering algorithm.

Require: Distance metric h; Radius ε; Minimum density n
Ensure: Clusters are formed
  for all points **s do**
    find core points through (2)
  end for
  for all core points **x do**
    find neighboring core points
  end for
  for all non-core points **s do**
    assign them to a cluster or mark them as noise
  end for
  return

Spectral clustering is based on the trajectories a random walker can take on a graph. Specifically, the normalized primary graph eigenvector contains the stationary distribution of the number of times the walker will visit each vertex. Therefore, it makes perfect sense to link the probability a vertex will be visited to its centrality. The eigenvector centrality is computed as in Eq. (3). At the k-th position the vector $\mathbf{g}$ contains the nonnegative centrality score for v_k. The existence of the eigenvalue with a value of one stems from the Perron-Frobenius theorem. Intuitively, the eigenvector represents a state of balance in the graph.

$$\mathbf{Ag} = \mathbf{g} \Leftrightarrow (\mathbf{I} - \mathbf{A})\mathbf{g} = \mathbf{0} \tag{3}$$

The above centrality metric comes from assuming a linear model for the centrality score g_k of vertex v_k as shown in (4) and it depends heavily on the topology of the underlying graph. Also under this model centrality is only additive and unweighted, while it relies exclusively on neighborhoods of depth one.

$$g_k = \sum_{v_i} g_i, \quad v_i \in \Gamma(v_k) \tag{4}$$

Recall that the neighborhood $\Gamma(v)$ of a vertex v with an edge set E is the set of vertices u adjacent to v as shown in Eq. (5). Consequently, the neighborhood cardinality $|\Gamma(v)|$ is by definition the degree of v. For directed vertices a distinction should be made for inbound and outbound neighborhoods.

$$\Gamma(v) \triangleq \{(u, v) \in E\} \tag{5}$$

A possible extension of (4) is through a nonlinear kernel $h(\cdot)\colon V \to \mathbb{R}^+$ which essentially changes how the centrality values coming from the neighborhood contribute to that of a given vertex. In this case the centrality is computed as in (6). Observe that the model is linear on the transformed centrality values $h(v_k)$.

$$h(v_k) \;=\; \sum_{v_i} h(v_i), \quad v_i \in \Gamma\,(v_k) \tag{6}$$

Another generalization of (4) is to assume a nonlinear model after the addition of the centrality values at each vertex. This model is similar to the function of most graph neural network (GNN) architectures in the sense that the nonlinear kernel is applied on the aggregation of the values from neighboring vertices and then the result is broadcast to neighboring vertices. Thus Eq. (4) essentially represents a steady state of the graph under a nonlinear model.

$$g_k \;=\; h\left(\sum_{v_i} g_i\right), \quad v_i \in \Gamma\,(v_k) \tag{7}$$

This leads to the nonlinear eigenvalue problem of (8). Therein the nonlinear function $\varphi(\cdot)$ is the result of $h(\cdot)$ to the linearly transformed variables $\mathbf{Ag}$, namely a composition of a nonlinear and a linear function. Note that in (8) the kernel $h(\cdot)$ is elementwise applied to $\mathbf{Ag}$ or, equivalently, the kernel $\varphi(\cdot)$ to $\mathbf{g}$.

$$\mathbf{g} \;=\; h(\mathbf{Ag}) \;=\; \varphi(\mathbf{g}) \tag{8}$$

Typically nonlinear eigenvalue problems are difficult to be solved. Consider for instance the delay eigenvalue problem of (9) and the quadratic eigenvalue problem of (10). Both have applications in many engineering scenarios and despite their simple closed form are difficult to be solved analytically in the general case. The quadratic eigenvalue problem of (9) has applications in control problems as well as in the stability of certain systems of differential equations.

$$\mathbf{M}(\lambda) \;=\; \mathbf{A}_0 + \lambda\mathbf{A}_1 + \lambda^2\mathbf{A}_2 \tag{9}$$

Another example of a difficult nonlinear problem is the delay eigenvalue problem of Eq. (10) which has applications among others to digital telecommunications, control theory, and acoustic systems. In Eq. (10) the positive integer p and the positive reals depend heavily on the underlying problem.

$$\mathbf{M}(\lambda) \;=\; -\mathbf{I}\lambda + \mathbf{A}_0 + \sum_{k=1}^{p} \mathbf{A}_k e^{-\lambda\tau_k} \tag{10}$$

Returning to (8) one way to solve the eigenvalue problem of $\varphi(\cdot)$ is to use a stationary point method. The latter is seeking through various means a point satisfying condition (11), which coincides exactly with the problem of (8).

$$\mathbf{x}^* \;=\; \varphi(\mathbf{x}^*) \tag{11}$$

A necessary condition for the stationary point methods of (11) is that the norm of the gradient of the function $\varphi(\mathbf{x})$ remains bounded. Essentially this curbs the fluctuations of the original function, ensuring thus higher local smoothness.

$$\|\nabla_{\mathbf{x}}\varphi(\mathbf{x})\|_2 \;\leq\; \beta_0 \tag{12}$$

Another condition is the regularization of (13) which controls the fluctuations of both the solution and the that of the q-th order derivative thereof. The latter has a relative weight of μ_0 which is a positive regularization hyperparameter (Fig. 3).

$$\|\varphi(\mathbf{x})\|_2 + \mu_0 \|\nabla^q_{\mathbf{x}}\varphi(\mathbf{x})\|_2 \;\leq\; \beta_1 \tag{13}$$

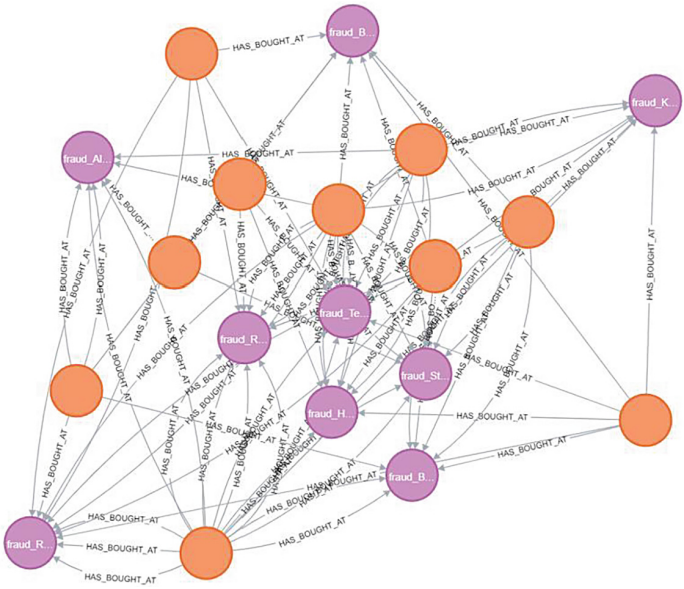

Fig. 3. Indicative transaction graph.

3.3 Classification

The third and final step of the proposed framework is the transaction classification. To keep the proposed framework simple, only the ubiquitous logistic regression has been used. The latter essentially partitions the decision plane to determine the respective probabilities of the outcomes of a binary decision. This is done under the assumption that the logit of the underlying distribution is a linear combination of the input variables as shown in Eq. (14).

$$\text{logit}\,(p) \;\triangleq\; \ln\frac{p}{1-p} \;=\; b_1x_1 + \ldots + b_mx_m + b_0 \;=\; \mathbf{b}^T\mathbf{x} \tag{14}$$

Logistic regression is frequently applied to a broad spectrum of problems ranging from financial engineering and smart city resources allocation to long logistic chains. This happens because of its reasonable power and simplicity.

4 Results

In this section the results of the proposed framework are presented and explained. The dataset used is the fraudulent transaction dataset available from Neo4j[1].

The results are given in Tables 4, 5, and 6 for the kNN, DBSCAN, and spectral clustering schemes.

Table 4. Results for kNN and logistic regression.

Fraudulent	Precision	Accuracy	F1 score
No	0.59	0.81	0.68
Yes	0.64	0.38	0.47

Table 5. Results for DBSCAN and logistic regression.

Fraudulent	Precision	Accuracy	F1 score
No	0.65	0.81	0.72
Yes	0.65	0.78	0.71

Table 6. Results for spectral clustering and logistic regression.

Fraudulent	Precision	Accuracy	F1 score
No	0.96	0.78	0.86
Yes	0.80	0.96	0.87

As is evident from the above tables, the spectral clustering combined with logistic regression gave the highest F1 score followed at some distance by DBSCAN, whereas kNN did not yield satisfactory results. This can be attributed to the fact that spectral clustering operates on the entire input graph fully utilizing its higher order structure. On the contrary, kNN is totally local relying solely on small neighborhoods, while DBSCAN lies in-between as it is based on larger segments of the data region and it is more parameterized (Figs. 4 and 5).

[1] https://neo4j.com/blog/fraud-detection/financial-services-neo4j/fraud-detection.

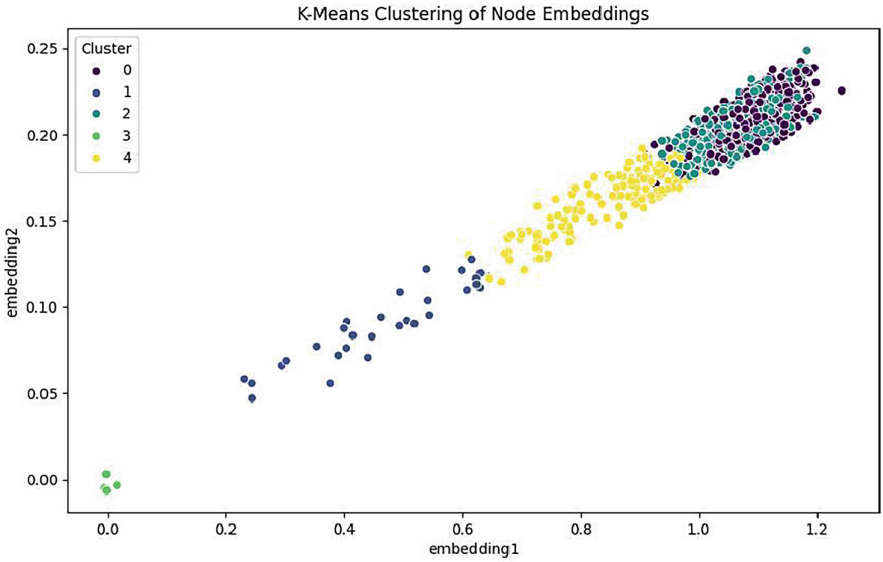

Fig. 4. Vertex clustering with kNN.

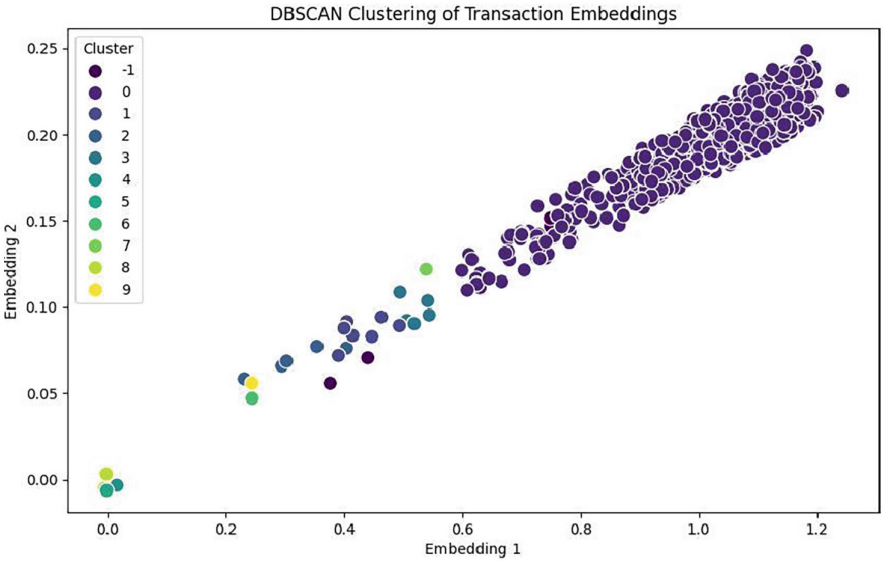

Fig. 5. Vertex clustering with DBSCAN.

5 Conclusions and Future Work

The focus of this conference paper is the discovery of fraudulent activities in credit card transaction graphs by exploiting their interconnected nature. Said discovery took place with performing vertex clustering and then performing logistic regression on the resulting partitioning. Each vertex contained an embedding of first order as well as higher order attributes. Graph clustering can identify suspicious clusters through hidden and latent patterns and facilitate fraud detection, resulting thus in a more robust digital economy. The proposed processing was implemented over Neo4j, a leading graph database. By representing credit card transactions, cardholders, and merchants as vertices and their relationships as edges, Neo4j enables the creation of a comprehensive graph model that captures the complex connections between these entities. Through the synergy of graph clustering and machine learning techniques provided by Neo4j the credit card fraud detection system displays considerable accuracy.

Regarding possible future research directions, a first step would be to test the proposed methodologies to larger transaction graphs which may well contain in their vertices more attributes. Additionally, the potential for real-time monitoring to provide protection against ever-evolving and new fraudulent patterns can be explored. Finally, sophisticated clustering or classification methodologies such as consensus clustering can be added to the proposed framework.

Acknowledgments. This conference paper is part of Project 451, a long term research initiative with a primary objective of developing novel, scalable, numerically stable, and interpretable higher order analytics.

The authors have no competing interests relevant to this work.

References

1. Ansel, J., et al.: PyTorch 2: faster machine learning through dynamic python bytecode transformation and graph compilation. In: International Conference on Architectural Support for Programming Languages and Operating Systems, vol. 2, pp. 929–947 (2024)
2. Bai, X., et al.: Construction of a knowledge graph for framework material enabled by large language models and its application. Comput. Mater. **11**(1), 51 (2025)
3. Berahmand, K., Mohammadi, M., Saberi-Movahed, F., Li, Y., Xu, Y.: Graph regularized nonnegative matrix factorization for community detection in attributed networks. IEEE Trans. Netw. Sci. Eng. **10**(1), 372–385 (2022)
4. Bhalekar, P.M., Saini, J.R.: Comprehensive exploration of the role of graph databases like Neo4j in cyber security. In: ESCI, pp. 1–4. IEEE (2024)
5. Bratanic, T.: Graph Algorithms for Data Science: With Examples in Neo4j. Simon and Schuster (2024)
6. Buchnik, I., Sagi, G., Leinwand, N., Loya, Y., Shlezinger, N., Routtenberg, T.: GSP-KalmanNet: tracking graph signals via neural-aided Kalman filtering. IEEE Transactions on Signal Processing (2024)
7. Chen, F., Cheung, G., Zhang, X.: Manifold graph signal restoration using gradient graph Laplacian regularizer. IEEE Trans. Signal Process. **72**, 744–761 (2024)

8. Cheng, J., Tang, Y., He, C., Feng, P., Han, K., Guan, Q.: Rethinking variational Bayes in community detection from graph signal perspective. Transactions on Knowledge and Data Engineering (2025)
9. Drakopoulos, G., Bardis, G., Mylonas, P.: Power iteration graph clustering with functools higher order methods. In: SMAP. IEEE (2024). https://doi.org/10.1109/SMAP63474.2024.00042
10. Drakopoulos, G., Giannoukou, I., Sioutas, S., Mylonas, P.: Self organizing maps for cultural content delivery. NCAA **31**(7) (2022). https://doi.org/10.1007/s00521-022-07376-1
11. Drakopoulos, G., Kafeza, E., Mylonas, P., Sioutas, S.: Approximate high dimensional graph mining with matrix polar factorization: a Twitter application. In: IEEE Big Data, pp. 4441–4449. IEEE (2021). https://doi.org/10.1109/BigData52589.2021.9671926
12. Drakopoulos, G., Kafeza, E., Mylonas, P., Sioutas, S.: Process mining analytics for Industry 4.0 with graph signal processing. In: WEBIST, pp. 553–560. SCITEPRESS (2021). https://doi.org/10.5220/0010718300003058
13. Drakopoulos, G., Kanavos, A.: Tensor-based document retrieval over Neo4j with an application to PubMed mining. In: IISA. IEEE (2016). https://doi.org/10.1109/IISA.2016.7785366
14. Drakopoulos, G., Mylonas, P.: A genetic algorithm for Boolean semiring matrix factorization with applications to graph mining. In: Big Data. IEEE (2022). https://doi.org/10.1109/BigData55660.2022.10020828
15. Drakopoulos, G., Mylonas, P.: Higher order probabilistic analysis of network trajectories of intelligent agents in Thespian. In: SEEDA-CECNSM. IEEE (2023). https://doi.org/10.1109/SEEDA-CECNSM61561.2023.10470503
16. Drakopoulos, G., Mylonas, P.: Clustering MBTI personalities with graph filters and self organizing maps over Pinecone. In: Big Data. IEEE (2024)
17. Drakopoulos, G., Voutos, Y., Mylonas, P.: Recent advances on ontology similarity metrics: a survey. In: SEEDA-CECNSM. IEEE (2020). https://doi.org/10.1109/SEEDA-CECNSM49515.2020.9221837
18. Fournier-Viger, P., et al.: Pattern mining: current challenges and opportunities. In: International Conference on Database Systems for Advanced Applications, pp. 34–49. Springer (2022)
19. Jamshidi, K., Xu, H., Vora, K.: Accelerating graph mining systems with subgraph morphing. In: European Conference on Computer Systems, pp. 162–181 (2023)
20. Kafeza, E., Drakopoulos, G., Mylonas, P.: Graph neural networks in PyTorch for link prediction in Industry 4.0 process graphs. In: AIAI. IFIP, vol. 713, pp. 220–234. Springer (2024). https://doi.org/10.1007/978-3-031-63219-8_17
21. Karavokyris, M., Sioutas, S., Drakopoulos, G., Kafeza, E.: Graph neural networks for affective social media: A comprehensive overview. In: CIKM Workshops (2022)
22. Koutra, D., Faloutsos, C.: Individual and collective graph mining: Principles, algorithms, and applications. Springer Nature (2022)
23. Lagraa, S., Husák, M., Seba, H., Vuppala, S., State, R., Ouedraogo, M.: A review on graph-based approaches for network security monitoring and botnet detection. Int. J. Inf. Secur. **23**(1), 119–140 (2024)
24. Li, M.M., Huang, K., Zitnik, M.: Graph representation learning in biomedicine and healthcare. Nat. Biomed. Eng. **6**(12), 1353–1369 (2022)
25. Li, Z., Lai, D.: Dynamic network embedding via temporal path adjacency matrix factorization. In: CIKM, pp. 1219–1228 (2022)
26. Lu, J., Nie, F., Wang, R., Li, X.: Fast multiview clustering by optimal graph mining. IEEE Transactions on Neural Networks and Learning Systems (2023)

27. Nguyen, L.B., Nguyen, L.T., Zelinka, I., Snasel, V., Nguyen, H.S., Vo, B.: A method for closed frequent subgraph mining in a single large graph. IEEE Access **9** (2021)
28. Reiser, P., et al.: Graph neural networks for materials science and chemistry. Commun. Mater. **3**(1), 93 (2022)
29. Şentürk, F., Aytac, V.: A graph-based ontology matching framework. N. Gener. Comput. **42**(1), 33–51 (2024)
30. Sharma, R., Meena, H.K.: Emerging trends in EEG signal processing: a systematic review. SN Comput. Sci. **5**(4) (2024)
31. Šikić, L., Kurdija, A.S., Vladimir, K., Šilić, M.: Graph neural network for source code defect prediction. IEEE Access **10**, 10402–10415 (2022)
32. Su, X., et al.: A comprehensive survey on community detection with deep learning. IEEE Transactions on Neural Networks and Learning Systems (2022)
33. Tay, D.B.: Dual frames of localized spectral graph wavelets. Digital Signal Processing (2025)
34. Tu, W., et al.: Attribute-missing graph clustering network. In: AAAI. vol. 38 (2024)
35. Veličković, P.: Everything is connected: graph neural networks. Curr. Opin. Struct. Biol. **79** (2023)
36. Wu, X., Lu, W., Quan, Y., Miao, Q., Sun, P.G.: Deep dual graph attention autoencoder for community detection. Expert Syst. Appl. **238** (2024)
37. Xie, H., Ma, J., Xiong, L., Yang, C.: Federated graph classification over Non-IID graphs. Adv. Neural. Inf. Process. Syst. **34**, 18839–18852 (2021)
38. Zang, Y., Ren, L., Wu, J., Xiao, Y., Hu, R.: Power on graph: mining power relationship via user interaction correlation. Expert Syst. Appl. (2025)
39. Zhang, F., et al.: OAG-Bench: a human-curated benchmark for academic graph mining. In: Conference on Knowledge Discovery and Data Mining, pp. 6214–6225. ACM (2024)
40. Zhang, F., Sun, B., Diao, X., Zhao, W., Shu, T.: Prediction of adverse drug reactions based on knowledge graph embedding. BMC Med. Inf. Decis. Making **21** (2021)
41. Zhang, J., Chen, Y., Liu, G., Gao, W., Li, G.: Efficient point cloud attribute compression framework using attribute-guided graph Fourier transform. In: ICASSP, pp. 8426–8430. IEEE (2024)
42. Zhang, M., Li, P.: Nested graph neural networks. Adv. Neural. Inf. Process. Syst. **34**, 15734–15747 (2021)
43. Zhang, Y., Zhao, H., Li, H., Chen, S.: FastMAC: stochastic spectral sampling of correspondence graph. In: Conference on Computer Vision and Pattern Recognition, pp. 17857–17867. IEEE/CVF (2024)