Mining Frequent Approximate Patterns in Large Networks

Kaouthar Driss[1], Wadii Boulila[1], Aurélie Leborgne[2], Pierre Gançarski[2]

[1]RIADI Laboratory, National School of Computer Sciences, University of Manouba, Tunisia

[2]ICube Laboratory, University of Strasbourg, France

Abstract

Frequent pattern mining algorithms often draw on graph isomorphism to identify common pattern occurrences. Recent research, however, has focused on cases in which patterns can differ from their occurrences. Such cases have great potential for the analysis of noisy network data. This approach can be refined still further, though. Most existing FPM algorithms consider differences in edges and their labels, but none of them so far has considered the structural differences of vertices and their labels. Discerning how to identify cases that differ from the initial pattern by any number of vertices, edges, or labels has become the main challenge in this approach. As a solution, we suggest a novel Frequent Pattern Mining (FMP) algorithm named Mining Frequent Patterns (MFP) with two central new characteristics. First, we begin by using the inexact matching technique, which allows for structural differences in edge, vertices, and labels. Second, we follow the approximate matching with a focus on mining patterns within the directed graph, as opposed to the more commonly explored case of patterns being mined from the undirected graph. Our results illustrate the effectiveness of this new MFP algorithm in identifying patterns within an optimized time.

Keywords: Frequent Pattern Mining; direct graph; undirected graph; approximate matching; networks; JSON noisy data

1. Introduction

Frequent pattern mining (FPM) is an analytical process that aims to discover recurring patterns and associations from data [1][2][3]. FPM is an important data mining technique used to extract meaningful, usable insights, and information from large volumes of data [4][5][6][7]. Moreover, FPM also describes the process of discovery uncovering frequent recurring sets of items in large graphs, as based on a frequency threshold specified by the user. Recent research has focused on many such algorithms with a focus on frequency thresholds used to identify frequent patterns [8] [9] [10] [11].

Frequent patterns can be mined from large graphs, either oriented or non-oriented, that model a specific type of a network (social, medical, vegetation, etc.)[12][13]. We note that a graph is a collection of nodes and edges with labels or attributes. An essential challenge for complex graphs is the detection of frequent patterns [14] [15]. FPM algorithms often use the graph isomorphism technique as a means of identifying pattern occurrences. However, in this work, we focused on the case where a pattern could differ from its occurrences, which can be essential to analyze noisy network data.

Let us suppose the case of simple modeling of some parts of the human brain, as depicted in Figure 1. We have a basic graph *(G*) and three patterns (*P1, P2*, and *P3*) mined according to

each node's occurrence in *G*. In this case, *G* has ten vertices and nine oriented edges. Each vertex appears *n* times in G (e.g., Striatum appears 3, Pallidum appears 2, and Mid-brain appears 1).

In the following example, let us fix the frequency threshold as follows: σ >=2. In other words, two will be the minimum number of occurrences needed for each vertex to be accounted for in the mining process.

Hence, in the following example, with σ >=2, Striatum vertex (P1), Pallidum vertex (P2), and Striatum and Pallidum vertices (P3) represent the set of frequent and approximate patterns mined from G as shown in Figure 1.

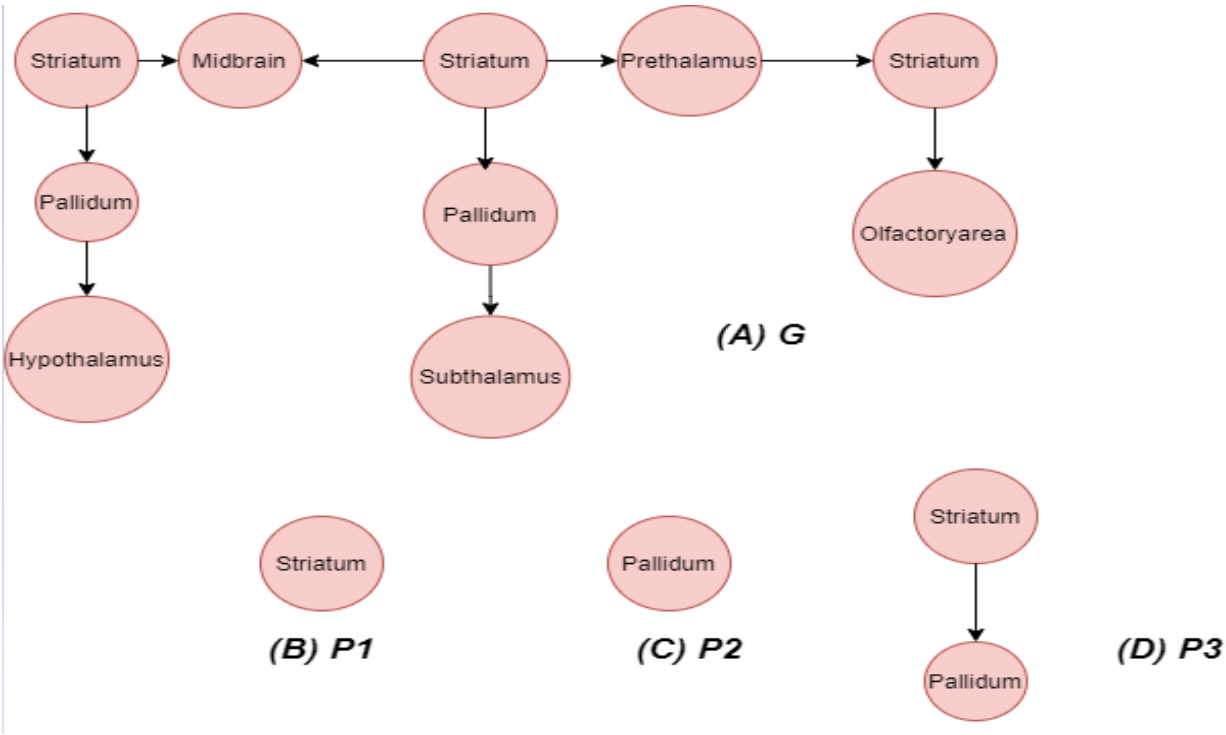


Figure 1. A basic example of a mined frequent and approximate set of patterns present on a graph G

Our work's primary challenge is detecting a set of frequent patterns using the inexact matching technique from a single oriented or non-oriented graph. Hence, we propose an approach based on three main steps. The first step, data processing, aims to prepare JSON noisy data by providing a more comprehensive and logical data structure. The second step, graph visualization, allows visualizing large graphs using built-in python libraries. The last step, mining frequent patterns, aims to discover recurring and approximate patterns using the proposed subgraph-mining function.

The remainder of our paper is organized in the following way. Background information on FPM is presented in Section 2. A review of the relevant literature is offered in Section 3. Our proposed FPM algorithm is introduced in Section 4, and our experimental results using it are detailed in Section 5. Our conclusions on applications [16][17][18] of this algorithm and our thoughts on directions for future work are the focus of Section 5.

2. Background

In the literature, several works have attempted to identify frequent patterns in a large graph through the use of Deep First Search (DFS) mechanisms [19] [20] and have focused on identifying recurring patterns following a pattern growth strategy [21].

In recent works, frequent approximate pattern mining is proposed as an extension of existing FPM algorithms [15][21][23][24][25]. A major challenge is to find approximate occurrences of patterns in single oriented and large graphs [20]. Finding patterns might be easy for graph isomorphism, but not for approximate mining patterns in a large graph. We need a frequency threshold to detect each vertex's number of times across the main graph for our search. This threshold is less computationally expensive, and the maximum number of occurrences of nodes represents the support of *P*. Given the existence of a graph (signified here by *G)* and a pattern within it (signified here by *P*), the threshold $\boldsymbol{\sigma}$ is defined by the equation (1) as follows:

$$\sigma\,(P,G) = max\,\{\,v \in V\,,\varphi(v)\} \tag{1}$$

V signifies the set of vertices of the graph *G,* and the function $\boldsymbol{\varphi}$ is the frequency count of each vertex v in *V.*

Furthermore, a function for graph comparison is required to identify patterns and ensure that the output covers only those patterns that are closed [18]. To accomplish this, we utilized a distance function to compute the dissimilarity between two patterns: *P1* and *P2*. The distance function, *fdist*, between *P1* and *P2* is demonstrated by equation (2).

$$fdis\,(P1,P2) = K\,.\mathrm{v} * \mathrm{select} + \,(1-\mathrm{K}).\mathrm{e} * \mathrm{select} \tag{2}$$

Where $0 \le K \le 1$ signifies the edit cost correlated with vertices that can measure the weight between vertices and edges as necessary, and $v * select$ and $e * select$ are defined by equations (3) and (4), respectively.

$$V * select = \sum_{v\,\in V1\setminus Rv1} dv\big(v, m(v)\big) + |Rv1| + |Rv2| \tag{3}$$

$$e * select = \sum_{(u,v)\,\in e1\setminus Re1} de((u,v),(m(u),m(v)) + |Re1| + |Re2| \tag{4}$$

Where

m is a bijective function that compares between vertices in *V1* and *V2*.

$d\big(v, m(v)\big)$: The cost of replacing *v* by $m(v)$.

$d\big((u,v),(m(u),m(v)\,)\big)$: The cost of replacing *(u,v)* by *(m(u),m(v))*.

RV1 is the set of vertices in *V1* that are unrelated to all vertices in *V2.*

RV2 is the set of vertices in *V2* that are unrelated to all vertices in *V1.*

RE1 is the set of edges related to *V1* that allow structural differences in edges.

RE2 is the set of edges related to *V2* that allow structural differences in edges.

3. Related works

Several studies from the literature have focused on FPM. In [9], researchers proposed and tested CFSP-Miner, a new form of closed frequent similar-pattern mining. Their results demonstrated that this new algorithm offered greater efficiency than similar, more frequently-used algorithms. Moreover, CFSP-Miner could locate a similar "closed" patterns without incurring significant information loss.

In [18], the authors surveyed the different mining algorithms presently designed to detect frequent sub-graphs within a single larger graph. They elaborated on which algorithms are most commonly used in the recent literature as well as which algorithms have impacted research on FSM processes. The authors also detailed the FSM process, which they split into its three primary phases: generation of candidates, computation of support, and achieving results.

Saidi et al. [23] have proposed the AntMot algorithm as a fast, alternative option for locating spatial motifs within three-dimensional protein structures. Such motifs are used in biological research as descriptors to aid with dataset classification. Their results demonstrated how the new approach offers significant enhancements in speed and accuracy of identifying sequential and frequent motifs.

Vella et al. [13] presented examples of biological application systems that utilized approaches based on graph theory. They introduced the reader to the protein-protein interaction (PPI) and co-expression networks, along with various aspects of reconstruction and analysis. The structure of PPI networks characterizes topological, functional, and disease modules.

In [20], the authors introduced two potential extensions of the previously mentioned AGraP algorithm, as a means of reducing output set sizes. A new variation of AGraP was proposed (CloseAFG) as well as a new algorithm (IntAFG). IntAFG followed a "greedy" approach and could be used as a stage post-mining stage in order to reduce the set of patterns. Experimental results demonstrated noticeable size reduction.

Mrzic et al. [21] offered a bioinformatics perspective in their survey of various subgraph-mining algorithms and their applications. The proposed approach focused on pattern discovery within large graphs. FSM techniques are utilized in bioinformatics to tackle a wide-ranging scope of research projects, from the discovery of frequent substructures in bimolecular compounds to the detection of understudied patterns in large-scale molecular networks.

In [7], the authors proposed a list-based, indexed algorithm developed to mine recent, high-utility patterns. The proposed algorithm reduced the utility values of past transactions based on the times at which data has been inserted. The researchers tested this method and compared its performance with state-of-the-art counterparts using various datasets, both real and synthetic. Their results demonstrated a higher commission for execution, memory usage, and overall scalability.

In [8], the authors performed a similarity analysis of sequential activity patterns frequently generated by various users. The researchers defined inter- and intra-personal measures of similarity between sets of two participants as well as among all of the patterns frequently created by a single individual. They also proposed an innovative new methodology framework to help complete this type of mining, an approach that they implemented on the dataset of the Safety Pilot project. This study was the first work completed utilizing BSM generated from connected vehicles.

In [14], the authors studied sequential pattern mining across large databases and developed a pattern-growth approach that would increase the efficiency and scalability of this process. The

method, which they named PrefixSpan, also included a pseudo-projection technique that would decrease the number of databases eventually generated. Experimental results demonstrated that the proposed PrefixSpan method often outperformed priority-based GSP algorithms such as FreeSpan, SPADE, and others, particularly in terms of speed.

In [10], the authors introduced the AGraP algorithm, which was developed to find frequent approximate patterns within a single graph. They also proposed a comparison function that could handle graphs based on edit distance and also accounted for structural differences within vertices and edges alike. They also suggested strategies that could be utilized to identify occurrences of structural differences in vertices. Here, the authors' results demonstrated that AGraP could find patterns identified by “gApprox” and also that the AGraP mining process depends mainly on which values the user selects for the frequency and dissimilarity thresholds.

In [4], the authors proposed a tree-based algorithm intended to mine temporal association rules by considering temporal relations among multi-items. The algorithm proposed could mine temporal associations in search of inter-transactions. By drawing on the T-FS-tree, candidate itemset generation could be avoided for mining rules, which effectively reduced the computational costs of scanning. With its basis in T-FS-tree structure, the new algorithm enabled the building of the tree and mining of the temporal relation proceeding simultaneously, which in turn improved both the efficiency of mining and the interpretability of results.

In [5], the authors compared the different algorithms available and most commonly used in Frequent Pattern Mining (FPM). They also reviewed the advantages and disadvantages of the most recent and significant FPM algorithms to support the development of more efficient FPM algorithms in the future. They reported that the major problem identified in this area concerns the hidden patterns frequently concealed within a data set, which often become increasingly time-consuming to mine as the amount of data to be mined increases.

In [26], the authors explained various methods of analyzing structures in social networks and proposed a hybrid method intended for the detection of overlapping communities within social networks. In the process they suggested, a new and optimized algorithm is introduced as a potential means of solving established optimization models. The authors also proposed a searchability enhancement involving a local search operator developed from the classic tabu search method. They tested the proposed algorithm and its effectiveness on various social networks. Series of experiments that yielded promising results in the detection of overlapping communities in such systems are conducted.

In [22], the authors proposed a specialized detection algorithm, or "for local higher-order community detection" (FuzLhocd). They also introduced a purely localized metric, the "local motif modularity," which was intended to limit the neighbourhoods around the seed node. Their experiments, which were conducted regarding both real-world networks and synthetic counterparts, proved that FuzLhocd could run efficiently. It could operate locally, and solve problems of seed dependency, all with a speed and accuracy that made it a highly effective and promising new tool.

In [24], the authors abstracted complex networks from various systems in which one vertex represents the individual, and the edge represents a relationship among such individuals. After applying their method to real world and artificial networks, the authors reported that greater scores for artificial networks and extrapolated that it would be used to great effect for overlapping or non-overlapping community structures networks.

In [25], the authors proposed an algorithm that could detect community structures within complex networks. Their experiments conducted on various benchmark networks indicated that

the algorithm being proposed was as effective as its more widely recognized, state-of-the-art counterparts.

4. The proposed approach

The proposed approach comprises three main steps: JSON data processing, data visualization, and mining frequent pattern (MFP). In the first step, we transform the input JSON data from unstructured to structured data. The second step entails data visualization, which deals with the visualization of large-real networks and basic graph simulation using Python packages such as NetworkX. The third step is dedicated to implementing of the MFP algorithm, which takes our large network/graph as input and returns a set of frequent approximate patterns.

Figure 2 depicts the three steps of the proposed approach.

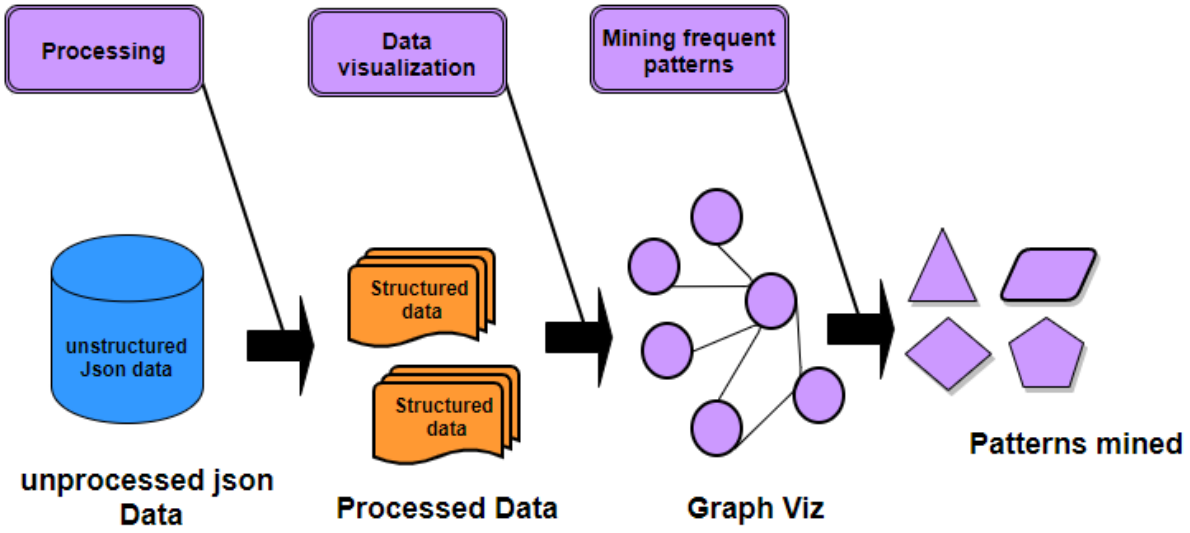


Figure 2. Main steps of the proposed approach

4.1. Data processing

This section presents the proposed data processing algorithm for JSON files needed to achieve the MFP algorithm and detect frequent approximate patterns in a single large graph.

This step aims to transform unstructured data into structured ones.

The mechanism of processing JSON files is depicted in algorithm 1. Here, a general script is applied to graphs stored in JSON files to give a more comprehensive and logical data structure. The nodes of the graphs represent geographical regions, brain regions, etc. Each node is labeled with the attribute “name” and identified by the attribute “id.” Edges or links can be labeled with a single attribute of "rel" or "type," or else doubly marked by two attributes: “rel” and “type.” The attribute “rel” refers to the name of the relationship between two nodes. Edges are identified by the nodes, or ids, of both its start and arrival vertices.

Algorithm 1 Processing of JSON files

Input: unstructured JSON file u_jf

Output: s_V, s_E

1: Import u_jf

2: Read u_V (v1, …, vn) // read nodes in u_jf

3: Read u_E (e1, …, en) // read edges in u_jf

4: s_V, s_E← ∅

5: Foreach v in u_V **do**

6: s_v← "v" + v.region + v.id

7: End foreach

8: Foreach e in u_E **do**

9: s_e← "e" + v.source + v.target + v.rel + v.type

10: End foreach

4.2. Data Visualization

The second step of our approach is data visualization. Using Python packages such as NetworkX, we can visualize large graphs, either oriented or non-oriented. We can also perform graph simulation to understand better and visualize primary networks.

In Figure 3, the graph is composed of eight vertices and seven edges. Each vertex has a name and a numerical identification number. For example, in our case, we considered bioinformatics data related to the human brain. Each node represents a region or a component of the human brain identified by an ID.

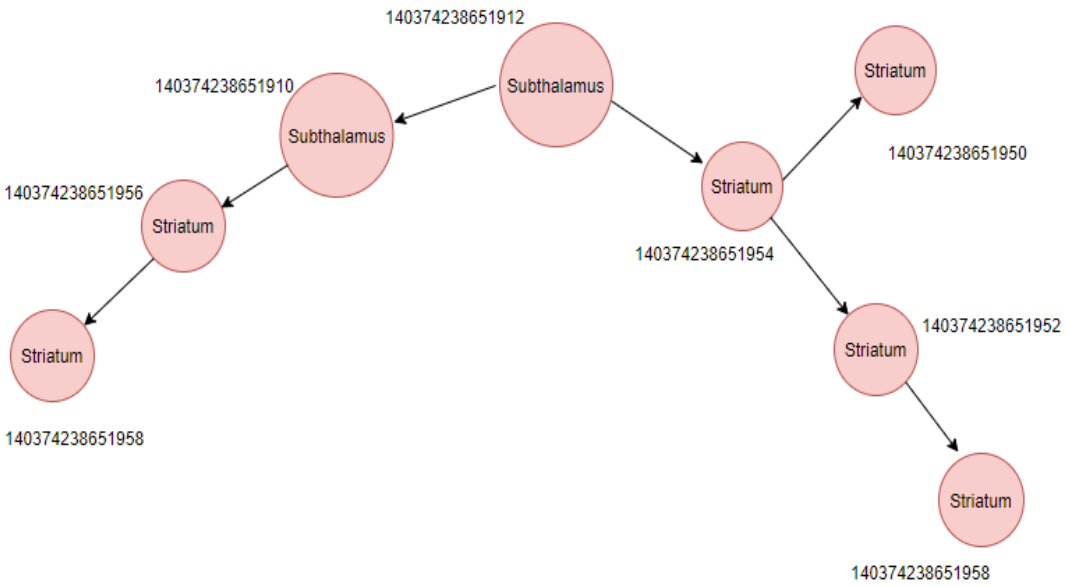


Figure 3. Example of a simple graph

4.3. Mining frequent patterns

The processes entailed in mining frequently occurring patterns is not a trivial problem. This becomes particularly difficult when patterns are extracted from large graphs [29][30][31][32]. Besides, graphs are a powerful modeling tool to represent the network's information structurally. In this context, the techniques we developed for the MFP algorithm are general, which can be applied to graphs from other domains, e.g., social networks [33], satellite images analysis [34], web analysis, etc. We present a new algorithm named MFP, which identifies frequent patterns in large graphs according to the input frequency threshold. The proposed MFP algorithm provides a set of frequent patterns using two combined techniques:

Inexact graph matching technique

We first utilize "inexact matching," which allows for the identification of structural differences among edges as well as vertices [26][27] [37][38]. In other words, pattern occurrences are not necessarily isomorphs since no bijective correspondence may be expected between them. This technique aims to search approximate occurrences of the frequent pattern in large graphs while considering differences in labels for both edges and nodes.

Oriented/ non-oriented graphs

The MFP algorithm includes finding a set of frequent approximate patterns in a directed graph having oriented edges from a vertex to one or many other vertices. Identifying recuring patterns approximately in an oriented graph is more rigorous than the identification process in the non-oriented graph since we avoid traversing paths between the graph‘s edges multiple times, which creates the risk of finding the same pattern many times [28] [37][39][40].

The main steps of the proposed MFP process are detailed in algorithm 2.

Algorithm 2 Mining frequent pattern in large graphs (MFP)

Input:

G (V, E)*:* Graph proposed for analysis

$\boldsymbol{\sigma}$: Frequency threshold

Mv: List of vertices with label equivalent to *v*

D: Dictionary that identifies and names equivalences between labels

Output: *P:* set of frequent patterns in graph (*G)*

1*:* Foreach $v \in V$ do

2: Stamp *v* as “explored”

3: **If** |Mv| >= $\boldsymbol{\sigma}$ **then**

4: Add *v* to *P*

5: Subgraph-Mining (*G*,*P*,*v*) ← Subgraph-Mining (*P*,*P*,*v*)

6: **End if**

7: End foreach

The subgraph-mining function is depicted in algorithm 3. This function is dedicated to finding the list of frequent patterns in the large graph from a small pattern considered an initial candidate specified as input. The subgraph-mining process takes the large graph *G* and the candidate pattern *P* as input, plus the node *Vnew* to add. The output provides a list of approximate occurrences (according to the labels of Vnew) of P.To do this, we create an empty list *Mp* and then we perform an exhaustive search in the large graph. In this search, we ensure that we consider the length of the path between the initial pattern *P* and the vertex *v* and its label which, should be approximately like *Vnew*'s label.

Function Subgraph-Mining

Input: G: graph for analysis

P: basic pattern candidate

Vnew: Vertex related to P

Output: *Mp*: list of pattern (*P*) occurrences = P ∪ Vnew

1: Create an empty list Mp

2: **Foreach** vertex v in G **do**

3: Calculate the maximum path length (*l)* between *P* and *v*

4 Obtain set of vertices connected to ***P*** through a path of length equal to or less than 1

5: Identify the subset ***Vl*** of vertices bearing label equivalent to label of *Vnew*

6: Add every graph P ∪ {v}, with v∈Vl added to new list Mp**7:**

8: **End foreach**

9: Return Mp.

The most common metric for calculating time complexity is "big O" notation. This removes all constant factors so that the running time can be estimated with *n* when *n* approaches infinity. Let us take the letter T as the time complexity of the MFP algorithm. MFP algorithm has time complexity equal to T(n) = O(n^2), which is acceptable compared to other recent algorithms.

5. Experiments

The experiments were run on a computer with an Intel Core i5 2.80 GHz (4CPUs), 8GB of RAM, 512GB SSD, and an Ubuntu 11 OS. The proposed algorithm was implemented using Python 3.6 (IDE Jupyter notebook and Pycharm).

5.1. Data Processing

Figure 4.A) represents the unstructured JSON data, which are stored in a list of dictionaries. Every single dictionary refers to one node or one edge of the large graph. For example, as shown in Figure 4.A), we have, in each dictionary, attributes related to each node, such as "name", "id", etc. The attribute name refers to the label of this node.The attribute time refers to the date of the initial image. The attribute region refers to the human brain or geographical region, and the attribute id is a combination of the two previous attributes (name and time).

In Figure 4.B), we referred to a node (vertex) of a graph using the letter 'v' and to edges (links) using the letter 'e'. As a result, we obtained two series: the first series of all vertices followed by the series of all edges. The node's arrangement, designed by the letter 'v', is followed by

the attribute region (region of the human brain or geographic area in remote sensing) and the attribute id.

The edge sequence is designed by the letter ‘e’ followed by “id” attribute of the starting node followed by the “id” attribute of the arrival node, relation attribute (RCC8 in remote sensing), and finally “type” attribute.

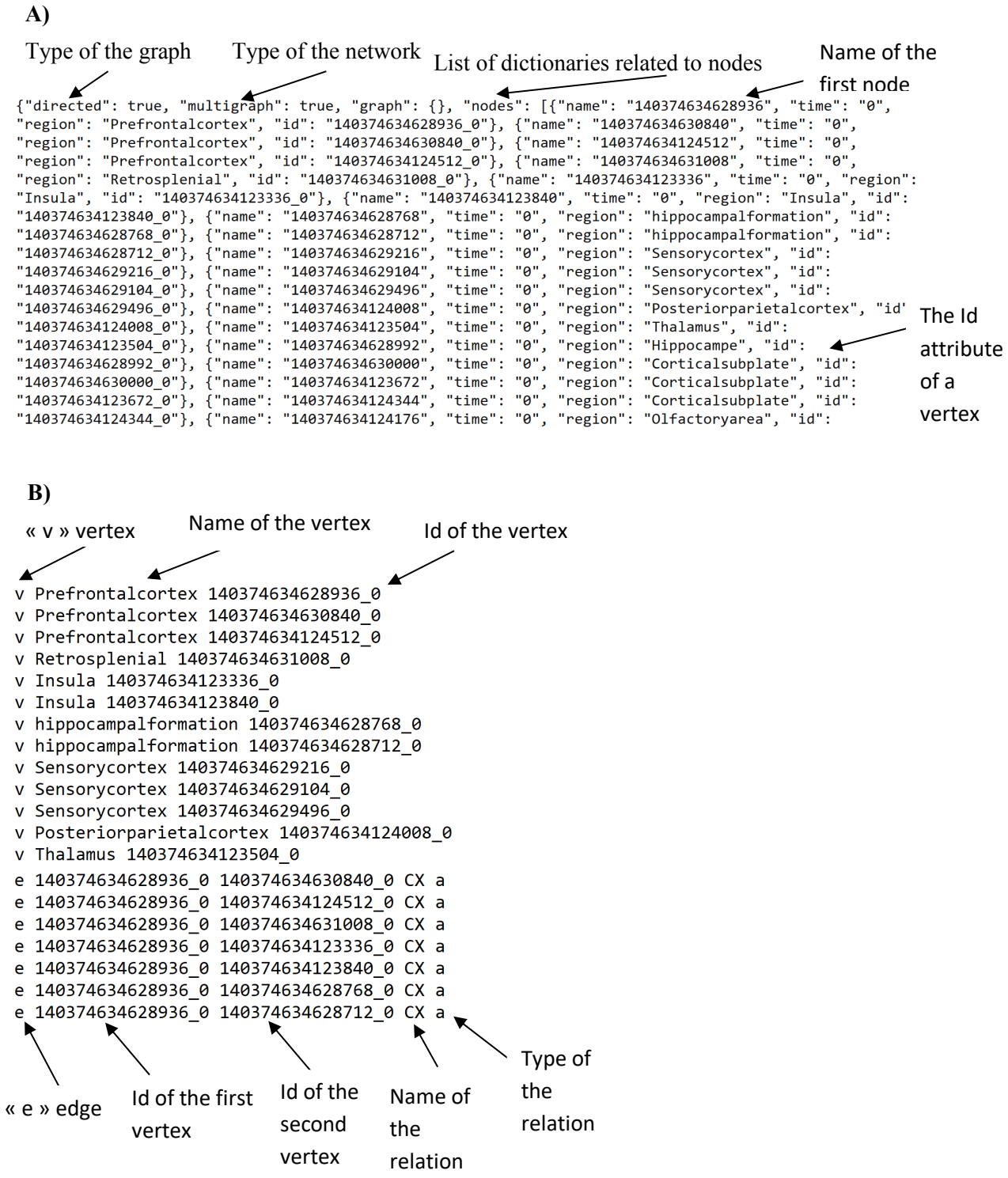


Figure 4. Processing of JSON Data: A) unstructured JSON data and B) structured JSON data

5.2. Data visualization

Figure 5 depicts a part of a sizeable oriented graph using the NetworkX python package. In this graph, vertices are labeled with Ids, and edges are oriented from one node to another.

NetworX is a very potent Python package developed to create, manipulate, and study both the structure and the dynamics of complex graphs modeling real networks.

NetworX also deals with many additional tasks, including the loading and storage of graphs in both standard and non-standard data formats as well as the generation of networks (both classic and random), the analysis of network structures, the construction of network models, the design of novel new network algorithms, the drawing of graphs, etc.

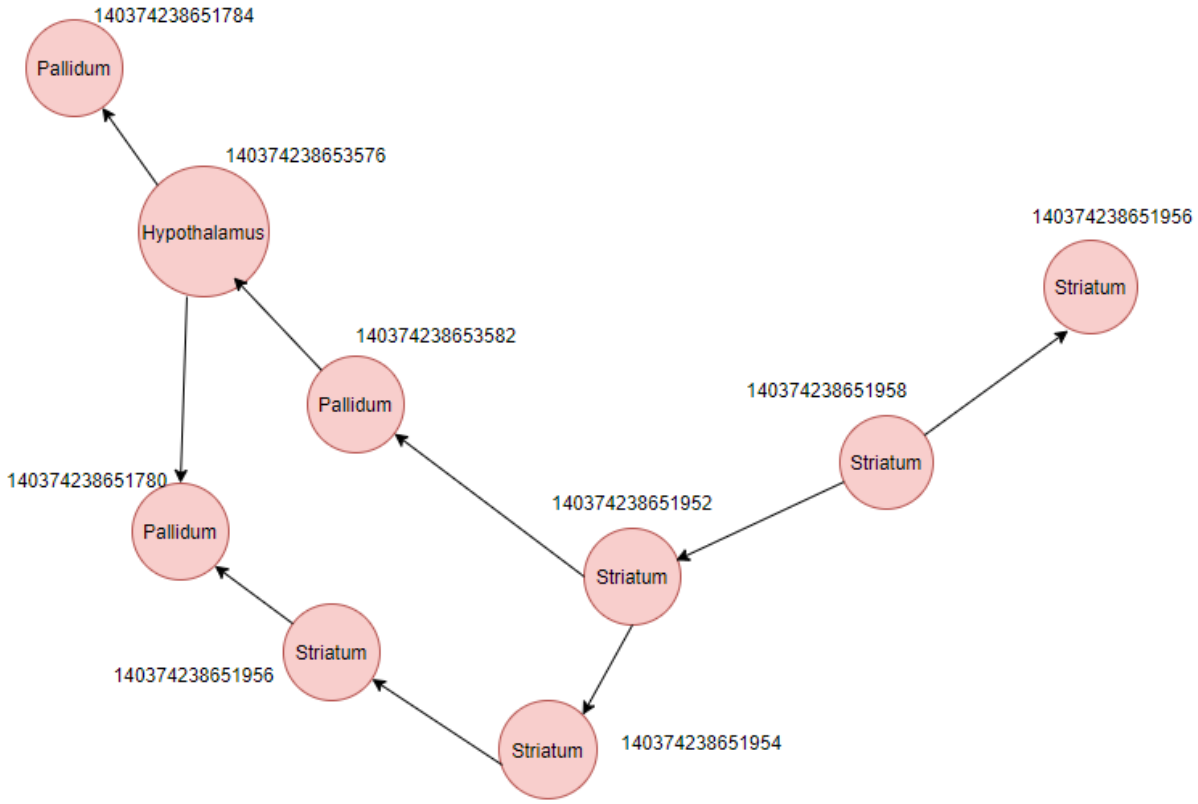
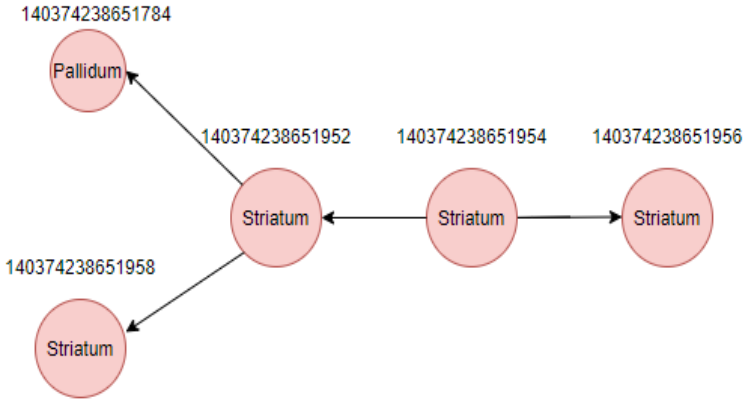


Figure 5. Visualizing a part of an oriented graph with the NetworkX package

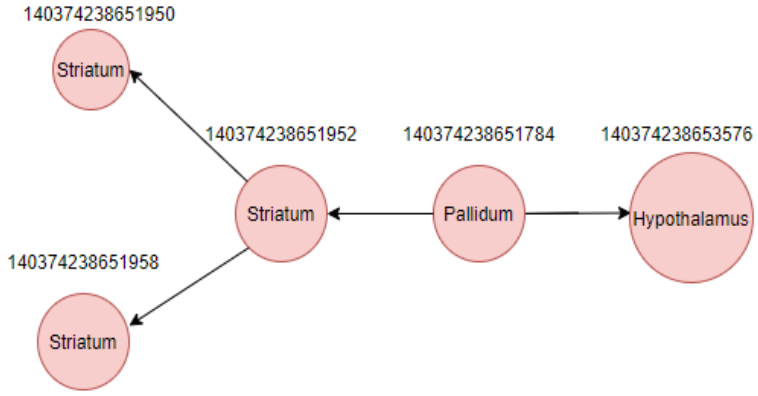
5.3. Mining frequent patterns

The proposed MFP algorithm identifies significant numbers of patterns using structural differences among edges, vertices, and labels. The MFP algorithm was implemented using Python.

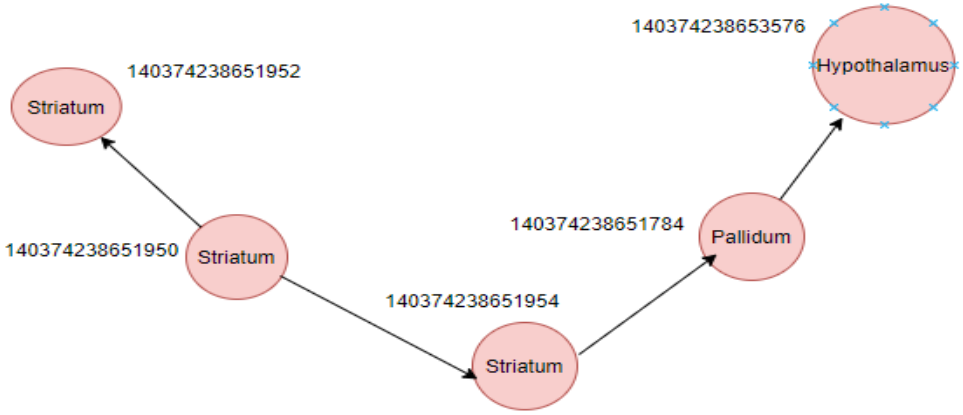
Figure 6 shows a set of approximate patterns with a lexicographical description. Figures 6.A) and 6.B) show frequent approximate patterns mined from a large graph where the pattern in Figure 6.B) is approximately the pattern in Figure 6.A). Figures 6.C) and 6.D) depict two other frequent approximate patterns in the same network. The Pattern in Figure 6.D) is approximately the more extensive occurrence of the pattern in Figure 6.C) with six oriented vertices.


140374238651784
Pallidum
140374238651952
140374238651954
140374238651956
Striatum
Striatum
Striatum
140374238651958
Striatum

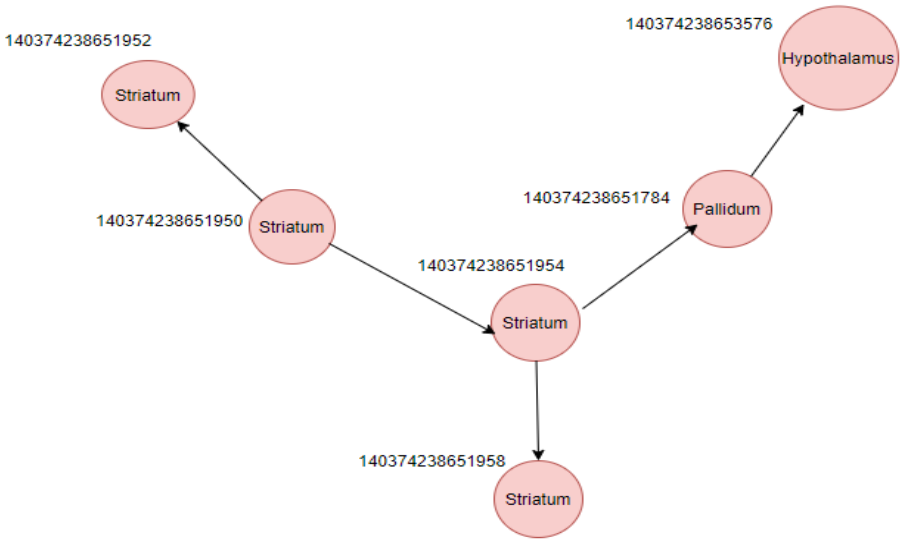
(A)


140374238651950
Striatum
140374238651952
140374238651784
140374238653576
Striatum
Pallidum
Hypothalamus
140374238651958
Striatum

(B)


140374238653576
Hypothalamus
140374238651952
Striatum
140374238651784
Pallidum
140374238651950
Striatum
140374238651954
Striatum

(C)



(D)

Figure 6. Overview of the main outputs: a set of approximate frequent patterns {A, B, C, D}.(A) Graphical description of the recurring primary pattern, (B) The approximate second recurring pattern of (A), (C) Third recurring approximate pattern, and (D)The most extensive frequent approximate pattern.

5.4. Experimental Evaluation

The following sections will compare the MFP and Gaston algorithms based on the three criteria: runtime, memory consumption, and the number of frequent mined patterns. The choice to compare with the Gaston algorithm is not random since the proposed algorithm MFP and Gaston share several common points, as depicted in Table 1.

Table 1. Comparison between MFP and Gaston algorithms

Feature	MFP algorithm	Gaston algorithm
Input type	Undirected/Directed graph	Undirected/Directed graph
Dynamicity of graphs	Static	Static
Result type	Frequent subgraphs	Frequent subgraphs
Pattern growth	Pattern growth-based approach	Pattern growth-based approach
Algorithmic design	Serial	Serial
Graph mining	Deep First Search (DFS)	Sequences and Tree Extraction (STE)

The comparison between MFP and Gaston algorithms will be based on two open molecular databases, as shown in Table 2. The NCI dataset is used to determine how an algorithm scale increases database size and contains 237771 graphs. In contrast, the HIV dataset contains both confirmed moderately active molecules (HIV CM) and confirmed active molecules (HIV CA). HIV dataset contains 42689 graphs.

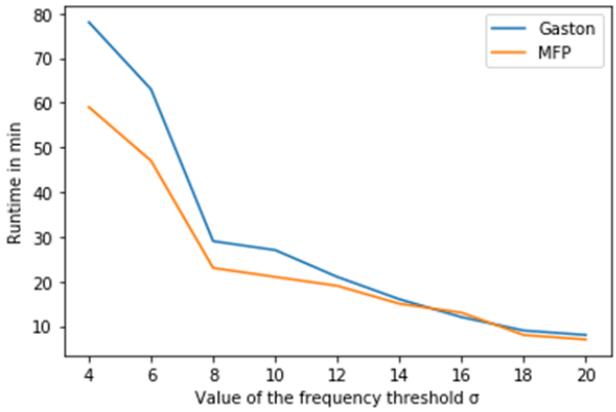
Table 2. Datasets used for comparison between MFP and Gaston algorithms

Dataset	Number of graphs	Field
NCI	237771	Chemical
HIV	42689	Chemical

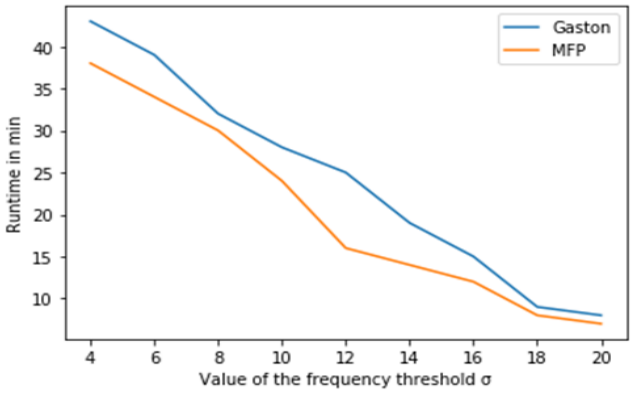
5.4.1 Runtime evaluation

MFP and Gaston algorithms' runtime is evaluated using graphs derived from the two datasets NCI and HIV. Since the two datasets include a significant number of graphs, the runtime will be calculated in minutes.

Figure 7 depicts a comparison of the runtime of MFP and Gaston algorithms for the two datasets NCI and HIV. Results demonstrate that the MFP algorithm outperforms the Gaston algorithm in terms of runtime for both datasets using different values of σ. The runtime varies according to the value of the frequency threshold σ. When we increase the frequency threshold value, the number of mined patterns decreases and the runtime becomes lower. This can be explained by the evidence that the more the frequency threshold is high, the less frequent and approximate patterns exist in graphs.



(A) NCI dataset



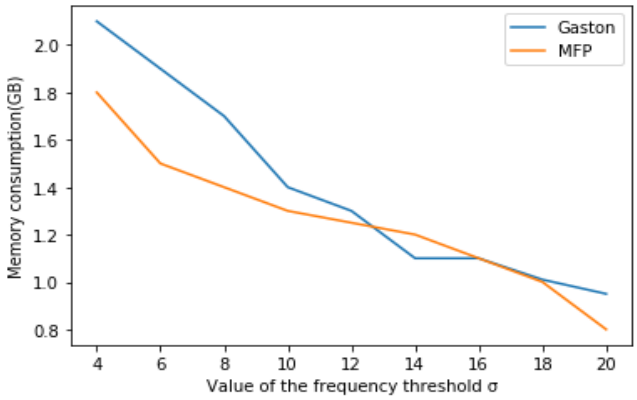
(B) HIV dataset

Figure 7. Comparison of the runtime between MFP and Gaston

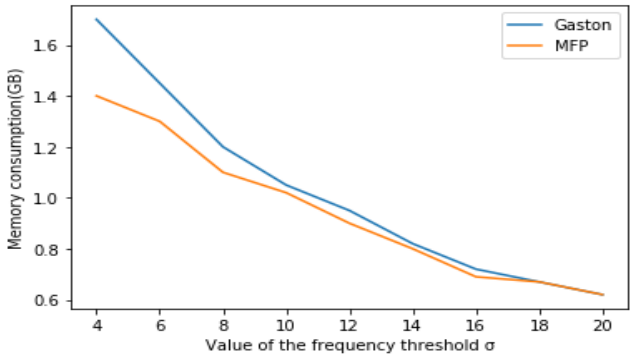
5.4.2 Memory consumption

This section will compare the memory usage in GB for MFP and Gaston algorithms using various values for σ.

Figure 8 depicts a comparison between MFP and Gaston algorithms according to memory usage for the NCI and HIV datasets. The comparison shows that the Gaston algorithm's memory usage is greater than the memory usage of the MFP algorithm for most cases of σ values. In other words, MFP is less expensive in terms of memory storage than the Gaston algorithm. This can be explained by the mining approach used by each of the two algorithms. MFP uses a deep first search (DFS) approach to find a frequent approximate list of patterns. However, Gaston uses sequences and tree extraction by integrating frequent paths as an approach.



(A) NCI dataset



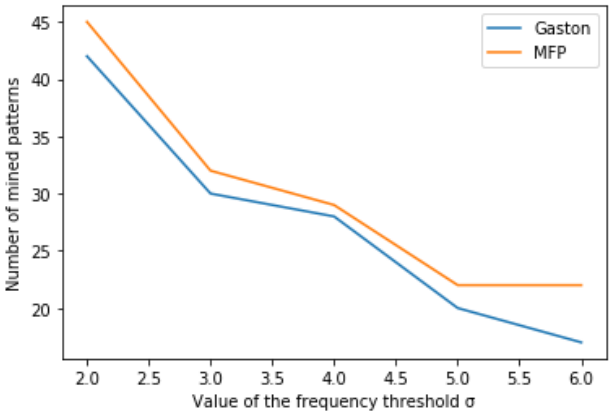
(B) HIV dataset

Figure 8. Comparison of memory consumption between MFP and Gaston

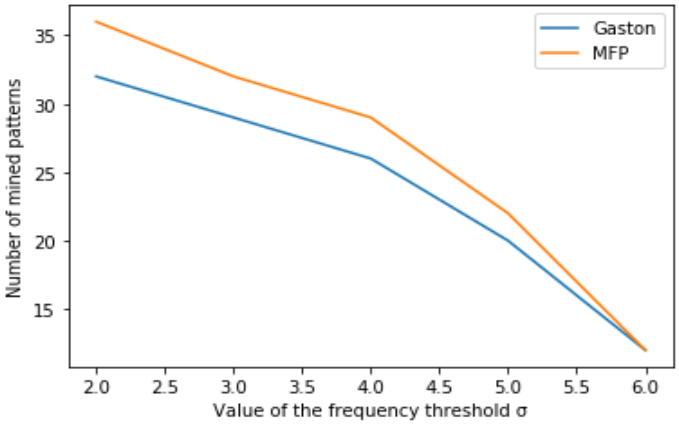
5.4.3 Number of frequent mined patterns

This section will compare the number of patterns mined by MFP and Gaston algorithms using various values for σ**.** Figure 9 shows that for the same value of σ, the number of patterns mined by the MFP algorithm is greater than that mined by Gaston. MFP can detect more frequent approximate patterns than Gaston. This last algorithm cannot detect all occurrences of a given P in G. For the NCI dataset, we noticed that the number of mined patterns becomes constant when the value of σ >=5; however, this number decreases for the Gaston algorithm when the value of σ increases. For the HIV dataset, both algorithms detect the same number of patterns when the value of σ becomes greater (σ>6). We can conclude that the MFP algorithm is more

efficient than the Gaston algorithm in terms of mining and finding more occurrences of a given pattern in a large graph.



(A) NCI dataset



(B) HIV dataset

Figure 9. Comparison of the number of frequent mined patterns between MFP and Gaston

In general, and based on the previous comparisons, it becomes evident that MFP is more efficient in runtime, memory usage, and the number of mined patterns in a large graph.

6. Conclusion

As data become more complicated, new algorithmic tools are needed to uncover frequent patterns. The identified patterns can be used to analyze associations, correlations, and other relationships between data, and to perform many extraction-related tasks, such as indexing, classification, etc. Hence, graph mining is being used with increasing frequency for analyzing large-scale networks with more complex data. In the paper above, we have presented a new approach based on the MFP algorithm, which can detect frequent patterns in large networks using the inexact matching technique. The proposed method is based on three steps: JSON data processing, data visualization, and mining frequent patterns using the MFP algorithm. Our algorithm was shown to be consistently more efficient than similar state-of-the-art algorithms when we tested it in real-world networks. Besides, we extended directed graphs to account for

the edge direction of large graphs. Experimental results on real-world data demonstrate that the proposed MFP algorithm provides good results compared to other cutting-edge methods.

Our work's primary ideas here also point toward several routes for the further, future development of optimized algorithms. Once frequent sets of patterns are found with the proposed MFP algorithm, additional analysis types could be applied, such as classification analysis, clustering, and building predictive models. Supervised machine learning algorithms could also be used for predictive analysis (e.g., Random Forest, neural networks, or a fusion of both).

References

[1] Han, J., Cheng, H., Xin, D., & Yan, X. (2007). Frequent pattern mining: current status and future directions. *Data mining and knowledge discovery*, *15*(1), 55-86.

[2] Bhuiyan, M. Z. A., Wu, J., Weiss, G. M., Hayajneh, T., Wang, T., & Wang, G. (2017). Event detection through differential pattern mining in cyber-physical systems. *IEEE Transactions on Big Data*.

[3] Chaki, J., Dey, N., Shi, F., & Sherratt, R. S. (2019). Pattern mining approaches used in sensor-based biometric recognition: A review. *IEEE Sensors Journal*, *19*(10), 3569-3580.

[4] Wang, L., Meng, J., Xu, P., & Peng, K. (2018). Mining temporal association rules with frequent itemsets tree. *Applied Soft Computing*, *62*, 817-829.

[5] Chee, C. H., Jaafar, J., Aziz, I. A., Hasan, M. H., & Yeoh, W. (2019). Algorithms for frequent itemset mining: a literature review. *Artificial Intelligence Review*, *52*(4), 2603-2621.

[6] Luna, J. M., Fournier-Viger, P., & Ventura, S. (2019). Frequent itemset mining: A 25 years review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *9*(6), e1329.

[7] Wang, C. S., & Chang, J. Y. (2019). MISFP-growth: hadoop-based frequent pattern mining with multiple item support. *Applied Sciences*, *9*(10), 2075.

[8] Nam, H., Yun, U., Yoon, E., & Lin, J. C. W. (2020). Efficient Approach of Recent High Utility Stream Pattern Mining with Indexed List Structure and Pruning Strategy Considering Arrival Times of Transactions. *Information Sciences*.

[9] Shou, Z., & Di, X. (2018). Similarity analysis of frequent sequential activity pattern mining. *Transportation Research Part C: Emerging Technologies*, *96*, 122-143.

[10] Rodríguez-González, A. Y., Lezama, F., Iglesias-Alvarez, C. A., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A., & de Cote, E. M. (2018). Closed frequent similar pattern mining: Reducing the number of frequent similar patterns without information loss. *Expert Systems with Applications*, *96*, 271-283.

[11] Flores-Garrido, M., Carrasco-Ochoa, J. A., & Martínez-Trinidad, J. F. (2015). AGraP: an algorithm for mining frequent patterns in a single graph using inexact matching. *Knowledge and Information Systems*, *44*(2), 385-406.

[12] Tamilmani, G., & Sivakumari, S. (2019). Early detection of brain cancer using association allotment hierarchical clustering. *International Journal of Imaging Systems and Technology*, *29*(4), 617-632.

[13] Jerlin Rubini, L., & Perumal, E. (2020). Efficient classification of chronic kidney disease by using multi-kernel support vector machine and fruit fly optimization algorithm. *International Journal of Imaging Systems and Technology*, *30*(3), 660-673.

[14] Fournier-Viger, P., Li, Z., Lin, J. C. W., Kiran, R. U., & Fujita, H. (2019). Efficient algorithms to identify periodic patterns in multiple sequences. *Information Sciences*, *489*, 205-226.

[15] Bhatia, V. (2018). *Efficient Pattern Mining of Big Data using Graphs* (Doctoral dissertation).

[16] Huang, Q., Hu, B., & Zhang, F. (2019). Evolutionary optimized fuzzy reasoning with mined diagnostic patterns for classification of breast tumors in ultrasound. *Information Sciences*, *502*, 525-536.

[17] Jerlin Rubini, L., & Perumal, E. (2020). Efficient classification of chronic kidney disease by using multi-kernel support vector machine and fruit fly optimization algorithm. *International Journal of Imaging Systems and Technology*, *30*(3), 660-673.

[18] Verma, A. K., Pal, S., & Kumar, S. (2020). Prediction of skin disease using ensemble data mining techniques and feature selection method—a comparative study. *Applied biochemistry and biotechnology*, *190*(2), 341-359.

[19] Vella, D., Zoppis, I., Mauri, G., Mauri, P., & Di Silvestre, D. (2017). From protein-protein interactions to protein co-expression networks: a new perspective to evaluate large-scale proteomic data. *EURASIP Journal on Bioinformatics and Systems Biology*, *2017*(1), 6.

[20] Mrzic, A., Meysman, P., Bittremieux, W., Moris, P., Cule, B., Goethals, B., & Laukens, K. (2018). Grasping frequent subgraph mining for bioinformatics applications. *BioData mining*, *11*(1), 20.

[21] Meng, T., Cai, L., He, T., Chen, L., & Deng, Z. (2019). Local Higher-Order Community Detection Based on Fuzzy Membership Functions. *IEEE Access*, *7*, 128510-128525.

[22] Petelin, B., Kononenko, I., Malačič, V., & Kukar, M. (2019). Frequent subgraph mining in oceanographic multi-level directed graphs. *International Journal of Geographical Information Science*, *33*(10), 1936-1959.

[23] Le, B., Huynh, U., & Dinh, D. T. (2018). A pure array structure and parallel strategy for high-utility sequential pattern mining. *Expert Systems with Applications*, *104*, 107-120.

[24] Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., ... & Hsu, M. C. (2004). Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on knowledge and data engineering*, *16*(11), 1424-1440.

[25] Kucherov, G., Salikhov, K., & Tsur, D. (2016). Approximate string matching using a bidirectional index. *Theoretical Computer Science*, *638*, 145-158.

[26] Caelli, T., & Kosinov, S. (2004). An eigenspace projection clustering method for inexact graph matching. *IEEE transactions on pattern analysis and machine intelligence*, *26*(4), 515-519.

[27] Flores-Garrido, M., Carrasco-Ochoa, J. A., & Martínez-Trinidad, J. F. (2018). Extensions to AGraP algorithm for finding a reduced set of inexact graph patterns. *International Journal of Pattern Recognition and Artificial Intelligence*, *32*(01), 1860012.

[28] Chen, H., Liu, M., Zhao, Y., Yan, X., Yan, D., & Cheng, J. (2018, April). G-miner: an efficient task-oriented graph mining system. In *Proceedings of the Thirteenth EuroSys Conference* (pp. 1-12).

[29] Zeng, J., Shao, C., Wang, X., & Miao, F. (2020, January). A new method for detecting communities in network based on the affinity between nodes. In *IOP Conference Series: Materials Science and Engineering* (Vol. 715, No. 1, p. 012024). IOP Publishing.

[30] Dhiman, A., & Jain, S. K. (2016, April). Frequent subgraph mining algorithms for single large graphs—A brief survey. In *2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Spring)* (pp. 1-6). IEEE.

[31] Zarei, B., & Meybodi, M. R. (2020). Detecting community structure in complex networks using genetic algorithm based on object migrating automata. *Computational Intelligence*, *36*(2), 824-860.

[32] Messaoudi, I., & Kamel, N. (2020). Overlapping community detection with a novel hybrid metaheuristic optimisation algorithm. *International Journal of Data Mining, Modelling and Management*, *12*(1), 118-139.

[33] Saidi, R., Dhifli, W., Maddouri, M., & Mephu Nguifo, E. (2019). Efficiently Mining Recurrent Substructures from Protein Three-Dimensional Structure Graphs. Journal of Computational Biology, 26(6), 561-571.

[34] Liaqat, M., Khan, S., Younis, M. S., Majid, M., & Rajpoot, K. (2019). Applying uncertain frequent pattern mining to improve ranking of retrieved images. *Applied Intelligence*, *49*(8), 2982-3001.

[35] Qiao, S., Han, N., Gao, Y., Li, R. H., Huang, J., Guo, J., ... & Wu, X. (2018). A fast parallel community discovery model on complex networks through approximate optimization. *IEEE Transactions on Knowledge and Data Engineering*, *30*(9), 1638-1651.

[36] Gabardo, A. C., Berretta, R., & Moscato, P. (2020). M-Link: a link clustering memetic algorithm for overlapping community detection. *Memetic Computing*, 1-13.

[37] Gan, W., Lin, J. C. W., Fournier-Viger, P., Chao, H. C., & Philip, S. Y. (2019). HUOPM: High-utility occupancy pattern mining. *IEEE transactions on cybernetics*, *50*(3), 1195-1208.

[38] Kucherov, G., Salikhov, K., & Tsur, D. (2016). Approximate string matching using a bidirectional index. *Theoretical Computer Science*, *638*, 145-158.

[39] Saradha, C. S., & Arul, P. (2019). AN OPTIMIZED OVERLAPPING AND DISJOINT COMMUNITY DETECTION TECHNIQUES USING IMPROVED COMMUNITY OVERLAP PROPAGATION ALGORITHM IN COMPLEX NETWORKS. *Journal of Critical Reviews*, *7*(4), 2020.

[40] Gadár, L., & Abonyi, J. (2019). Frequent pattern mining in multidimensional organizational networks. *Scientific reports*, *9*(1), 1-12.