Survey on Isomorphic Graph Algorithms for Graph Analytics

Theyvaa Sangkaran[1]†, Azween Abdullah[2]††, N.Z. JhanJhi[3]†††, Mahadevan Supramaniam[4]††††

School of Computing and IT (SoCIT), Taylor's University, Malaysia
Research and Innovation Management Center , SEGI University, Malaysia

Summary
The social network analysis focuses on investigating the connection of social actors and obtaining useful insights. It involves various techniques and methods, including determining the isomorphism among (sub)graphs. In this study, we have presented a critical analysis of seven different algorithms related to graph analytics, namely, Ullman, VF, VF2, MLC, Schmidt and Druffel, Corneil and Gotlieb and canonical graph labelling, by reviewing different types of isomorphic graphs identification algorithms. This research contributes to the domain of knowledge by providing the working mechanisms, features, different application areas and mechanism states of various algorithms used for isomorphic graph.

Key words:
Algorithm for graph isomorphism, Social Network, Cyber Security, Graph analytics, Isomorphic graph.

1. Introduction

The social network analysis (SNA) elicited a remarkable attention among researchers who are participating actively in this domain [1], focusing on graph theory and social sciences [2]. Social science is the science that investigates the interconnection among objects or entities. Meanwhile, graph theory, a mathematical concept [3], provides the representation of objects and their corresponding interconnections as a vertices and edges. SNA is the science of analysing and/or visualising the vertices and edges of any social structure, including, but not limited to, social structure derived from various online media social platforms, academic relations (e.g. students, researchers and publications), criminal network and extended family ties.

Meanwhile, graph theory provides the foundational mathematical concept for structural representation of entities and their corresponding interactions in providing a network [4]. In addition, it is an enabling factor for the successful analysis of graphs in combination with other analytical tools and methods. A graph is a diagram that illustrates the relationships among variable quantities. The variable quantities have several terms, such as vertex/vertices, node, site and agent/actor, whereas the relationships are often referred to as bond, link/connection, edges or relational tie depending on the field of study [5].

Manuscript received January 5, 2019
Manuscript revised January 20, 2019

Figure 1 shows the basic example of a graph, whereas Table 1 provides the name of its components in different fields of study

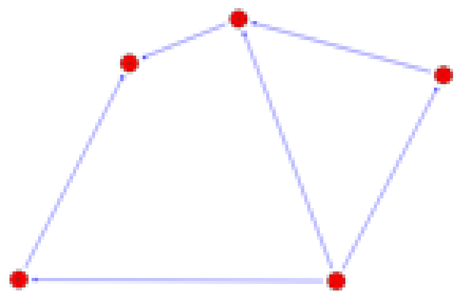
Fig. 1 Example of a graph (node in red and link in blue)

Table 1: Graph variable quantities in different fields of study.

Computer Science	Mathematics	Physics	Sociology	Computer Science
Node	Vertex /vertices	Site	Agent/actor	Node
Link /connection	Edge	Bond	Relational tie	Link /connection

Graph analytics (GA) involves all methods or techniques and is a tool used to understand, codify and visualise relationships that exist among people or devices in a network [6]. GA is constructed on the mathematics of graph theory and is used to model pairwise relationships between people, objects or nodes in a network. Currently, it is mainly employed to uncover insights regarding the strengths and directions of relationships. This analytics remarkably assists in obtaining solutions to problems inherent to a network structure. Graphs have different types (which will be discussed in the later sections of this paper), but unique to this research are 'isomorphic' graphs. By definition, two graphs T1 and T2 are regarded as isomorphic if (a) the number of components contained in each graph is equal and if (b) the edge connectivity is maintained [7]. Moreover, the isomorphic status of two or more subgraphs can be determined using these criteria. The discovery of isomorphic graphs is usually performed by running algorithms to analyse the graph structure by considering the above-mentioned conditions. Basically, an isomorphic algorithm is a graph-matching method. The importance of identifying isomorphic graphs is related to determining and linking similar graphs or subgraphs among social networks or even within a social network. A thorough review of the algorithms used in determining the isomorphic status of graphs is the main objective of this research.

This paper aims at achieving an in-depth understanding of the used algorithms and their behaviour. Section II presents the literature review of graph theory and isomorphic graph algorithms. Section III discusses the applications and our findings. Finally, Sections IV and V provide the discussion and conclusion of the study, respectively.

2. Literature Review

This section contains the comprehensive review of graph and its types, graph theory and its components and isomorphic graph algorithms in sequence.

2.1 Graph and Its Types

A graph [8] is considered to be an ordered pair G = (N, L) if it has a set of N nodes with a corresponding set of L links, which are subsets with two elements of N (i.e. a link is associated with two nodes, and that association uses the form of an unordered pair composed of two vertices). However, in strict distinction, graphs have various types depending on different degrees of generality [9]. The types of graphs are discussed in the following sub-sections:

Undirected Graphs: This is a type of graph [10] that lacks orientation, that is, edge (a, b) is identical to edge (b, a). Undirected graphs consist of unordered pairs of two nodes. The maximum possible number of edges in an undirected graph is n (n–1)/2 (without a loop) [11].

Directed Graph: This graph is composed of edges with orientations, usually written as an ordered pair G = (V, E) [12]. Directed graph (or digraph) is also regarded as symmetric if the corresponding inverted arrow also belongs to G for every arrow in G. The annotation V is a set whose elements are called vertices, nodes or points and annotation E is a set of ordered pairs of vertices, called arrows or directed (edges, arcs or lines). Moreover, a directed acyclic graph (or DAG) G is a digraph with no directed cycles [13].

Mixed Graph: This type of graph consists of directed and undirected edges for a given set of vertices, written as an ordered triple G = (V, E, A). It contains an order pair usually called s arc and an unordered pair called the edge of a graph [14]. Specifically, this graph type contains oriented and non-oriented edges. Mixed graph represents the physical interactions that include tasks performed in one and/or two different directions, often used in areas such as physical science, engineering, communication technology and many others to simulate certain problems [15].

Simple Graph: A simple graph is similar to an undirected graph without multiple edges or loops [16]. In a simple graph, each edge is an unordered pair of distinct vertices and the degree of every vertex is at most n–1. In a simple mathematical representation, a graph G = (V, E) consists of V, indicating a nonempty set of vertices, and E, which is several unordered pairs of distinct edges [17].

Multigraph: This graph is typically an undirected one but with multiple edges and possibly with loops [18]. Multiple edges are two or more edges that connect the same two vertices, where a loop is an edge that connects a vertex to itself (only permitted based on the application domain). A multigraph may exist as either oriented (a graph whose edges have arrows, indicating the starting and ending points of the edges) or non-oriented graph [19].

Directed Multigraph: This graph is the type that consists of a set of vertices V, edges E and function f from E to $\{\{u, v\} : u, v\ V\}\}$. The edges e1 and e1 are regarded as multiple edges if f(e1) = f(e2) [20].

Weighted Graph: This type of graph contains edges that have assigned weights, representing cost, strength, value, length or capacities depending on the domain application type. Often times, this graph type is referred to as a network.

Pseudograph: This graph consists of set of vertices V, edges E and function F from E to $\{\{u, v\} : u, v\ \dot{\mathrm{I}}\ V\}\}$. Loops are also allowed in this graph type.

The tabular breakdown of various types of graph is presented in Table 2 (where the features for each graph are highlighted).

Table 2: Summary of graph types and its uniqueness

Types	**Edges**	**With multiple edges**	**With loops**
Simple graph	Undirected	No	No
Undirected graph	Directed	Yes	No
Directed graph	Directed	No	Yes
Pseudograph	Undirected	Yes	Yes
Multigraph	Undirected	Yes	No
Directed multigraph	Directed	Yes	Yes

In addition, Table 3 presents the representation of graph and its components according to various fields of study.

Table 3: Graph types according to various fields of study

Graphs	Vertices	Edges
Chemical Compounds	Molecules	Bonds
Circuits	Gates, registers, processors	Wires
Transportation	Street intersections, airports	Highways, airway routes
Communication	Telephones, computers	Fibre optic cables
Mechanical	Joints	Rods, beams, springs
Internet	Web pages	Hyperlinks
Neural networks	Neuron	Synapses
Games	Board positions	Legal moves
Financial	Stocks, currency	Transactions
Protein networks	Proteins	Protein–protein interactions
Scheduling	Tasks	Precedence constraints

2.2 Graph Theory and Its Basic Components

The graph theory has been extensively used in social network analyses owing to its representational capacity and simplicity. Basically, all graphs consist of nodes (N) and connections (C), which connect the nodes (except for an empty set) [21]. The following are the basic components of the graph theory:

Dyad: It is regarded as the simplest network form, with only two nodes, which can either be connected or not. A connected dyad represents the property of a pair.

Triad: Any graph network formed by three nodes, with possible connection among the nodes, is referred to as triad. This network has several advantages, such as transitivity and equilibrium. A triad has also some basic conditions; for example, the maximum possible dyad in a triad is three.

Group: It is a set of nodes and their corresponding connections that have a defined limit. The limitation set for this network type defines the group such as a study group of students.

Sub-group: It refers to a branch of nodes and connections found within a group. A sub-group is composed of dyads and triads, but with a specific trait. It is useful when analysing a large social network group for cluster identification.

Relationship: It is simply an interaction, link, connection or edge between a set of nodes (at least two nodes). Relationship aids in understanding a network and shows the interaction of nodes.

2.3 Isomorphic (Sub)graph Algorithms

The process of matching two graphs G1 = (N1, B1) and G2 = (N2, B2) involves determining the mapping M associated with the nodes of both graphs. Different constraints can be placed on M, thus, leading to various mapping types [11], such as monomorphism, strict isomorphism, automorphism and graph–subgraph isomorphism. The most important path aspect of graph processing is graph matching for direct comparison [22] and an exact graph matching is referred to as an isomorphism problem [23]. Isomorphic (sub)graph algorithm simply accepts as input and a pair of directed graphs and returns 'true' when one graph is simply a 'relabelling' of the other. Various algorithms used to determine whether a graph is isomorphic or not are reviewed in the following sub-section:

Ullman Algorithm: The Ullman algorithm for subgraph isomorphism was developed compare with that of Corneil and Gotlieb because it does not process two graphs separately while determining the isomorphism [24]. The algorithm was also developed in two sequences; the second one is cited in this paper (because it is the refined form of the first algorithm developed by Ullman).

The refined procedure for the algorithm is as follows:

```
START
Step 1:  $M = M°, d = 1, H1 = 0;$
            FOR all $I = 1; ..., p_α$ set $f_i = 0;$
            refine M, if exit FAIL then
            TERMINATE algorithm;
Step 2:  IF there is no value of j such that $m_{dj} = 1$ and $f^j$
            $= 0$ THEN GO TO step 7,
            $Md = M;$
            IF $d = 1$ then $k = H1$ else $k = 0;$
Step 3:     $k = k + 1$
            IF $m_{dk} = 0$ or $f_k = 1$ then go to step 3;
            FOR all $j ≠ k$ set $m_{dj} = 0;$
            REFINE M; if exit FAIL THEN GO TO
            step 5;
Step 4:     IF $d < p_α$ THEN GO TO step 6 else give
            OUTPUT to indicate that an isomorphism
            has been found;
Step 5:     IF there is no $j > k$ such that $m_{dj} = 1$ and $f_i$
            $= 0$ THEN GO TO step 7,
            $M = Md;$
            GO TO step 3,
Step 6:     $H_d = k; F_k = 1; d = d + 1;$
            GO TO step 2,
Step 7:     IF $d = 1$ then TERMINATE algorithm,
            $F_k = 0; d = d–1; M = M_d, k:= H_d;$
                          GO TO step 5;
    END
                                          (1)
```

The algorithm above is an excerpt from [13].

Reportedly, the refinement procedure converges in a finite number of steps because it does not change 0 to 1 in M and the number of 1's in M is finite. [13]
VF: VF algorithm is another method for graph matching (isomorphic) and identifying isomorphic graphs or subgraphs in this case. This algorithm is presented below. As stated, the matching process between two graphs G1 = (N1, B1) and G2 = (N2, B2) involves in the determination of the mapping M associated with the nodes of the graph G1 to the nodes of G2 and vice versa [25].

```
    Algorithm:
PROCEDURE   Match (s)
        INPUT: an intermediate state s; the initial state
               $s_o$ has $M(s_o) = \Theta$
        OUTPUT: the mappings between the two graphs


        IF M (s) covers all the node THEN
               OUTPUT M (s)
    ELSE
              Compute the set P(s) of the pairs candidate
              for inclusion in M (s)
    FOREACH p ε P (s)
              IF the feasibility rules succeed for the
              inclusion of p in M (s)
    THEN
              Compute the state s' obtained by adding p
              to M (s)
                 CALL Match (s')
                 END IF
                 END FOREACH
                    END IF
  END PROCEDURE
```

The algorithm above is an excerpt from [25].
VF2: This graph isomorphism algorithm is based on the VF algorithm, with only slight changes, which can be observed in the algorithm structure as follows [26]:

```
Algorithm:
    PROCEDURE Match (s)
              INPUT: an intermediate state s; the initial
              state $s_o$ has $M(s_o) = \Theta$
              OUTPUT: the mappings between the two
              graphs

              IF M (s) covers all the node of $G_2$ THEN
                 OUTPUT M (s)
                    ELSE
              Compute the set P(s) of the pairs candidate
              for inclusion in M (s)
              FOREACH p ε P (s)
              IF the feasibility rules succeed for the
              inclusion of p in M (s)
    THEN
              Compute the state s′ obtained by adding p
              to M (s)
                 CALL Match (s′)
                    END IF
                 END FOREACH
                    Restore data structures
                    END IF
END PROCEDURE
```

The algorithm above is an excerpt from [26].

Merging Lexicographic Chain: Based on the constraint of a graph with unique node labels, this algorithm matches with the sorted adjacency lists by using the method closely related to merging sort (where its name is derived) [27]. The algorithm is tedious, with two steps having long internal processes. Throughout the algorithmic process, G represents the main graph, whereas SG represents the query graph. Succinctly, the first step involves the preprocessing, including the construction of adjacency list of nodes with all node properties. The second step provides the adjacency list comparison (i.e. find the match of the lexical chain list of the query graph with that of the model graph, and then return 'Matching is found' if and only if all the nodes in the query graph is found).
Schmidt and Druffel: According to the authors in [28], this algorithm also assesses a pair of digraphs for isomorphism by utilising the backtracking algorithm. By using the information in the distance matrix representation of a graph, this algorithm establishes the initial partition of the graph's vertices and reduces the search tree mapping possibilities by using the backtracking procedure. The algorithmic procedure is stated as follows:

(Initial portioning algorithm) [28] – PART 1

```
        START
1. SET $C_{t1} = 0\ (1 \le t \le N)$
2. COMPUTE the row and column characteristic matrices,
   $XR^1$ and $XC^1$ and $G^1$ and $XR^2$ and $XC^2$ for $G^2$
   $XR^1 = \{xr_i^{1m} \mid (1 \le i \le N) \wedge (1 \le m \le N-1) \wedge xr^1_{im} = \mid d^1_{ij} \mid (1 \le j \le N) \wedge (d^1_{ij} = m)\} \mid\}$
   $XR^2 = \{xr_s^{2m} \mid (1 \le s \le N) \wedge (1 \le m \le N-1) \wedge xr^2_{sm} = \mid d^2_{st} \mid (1 \le t \le N) \wedge (d^2_{st} = m)\} \mid\}$
   $XC^1 = \{xc_i^{1m} \mid (1 \le j \le N) \wedge (1 \le m \le N-1) \wedge xc^1_{jm} = \mid d^1_{ij} \mid (1 \le i \le N) \wedge (d^1_{ij} = m)\} \mid\}$
```

```
$XC^2 = \{xc_t{}^{2\,m} \mid (1 \le t \le N) \wedge (1 \le m \le N-1) \wedge xc^2_{tm} = \mid d^2_{st} \mid (1 \le t \le N) \wedge (d^2_{st} = m)\} \mid\}$
3. COMPOSE $XR^1$ with $XC^1$ to form $X^1$ and $XR^2$ with $XC^2$ to form $X^2$
4. CLASS = 0
5. CLASS = CLASS + 1
6. SELECT some integer $k \in \{1,2, ..., N\}$ such that $c1k = 0$ If none, STOP
7. Determine two set of integers
   $W1 = \{i \mid x1tm = x1km\ (1 \le m \le N-1)\}$ $W2 = \{i \mid x2tm = x1km\ (1 \le m \le N-1)\}$
8. Make class assignments:
   $Ct1$ = CLASS, $i \in W1$        $Ct2$ = CLASS, $i \in W2$
9. GO TO 5
   END
```

(Algorithm 2: Backtracking algorithm to test G^1 and G^2 for isomorphism) – PART 2

```
START
1. SET $t\ 1 = 0$     Set $p_i = 0\ (0 \le i \le N)$ ($p_t$ is the vertex in Graph $G^1$ chosen at level t)
2. LET $C^1_0$ and $C^2_0$ be the class vectors from Algorithm 1 and let $K^1_t$ and $K^2_t$ be the class count vector
3. IF $t = N$, conclude that the graphs are isomorphic, present the mapping and STOP
4. Set $i = pt$
5. IF $t = N$, conclude that the graph is isomorphic, present the mapping and STOP
6. CHOOSE some integer i for which $u^1_u$ has not been mapped at a lower level If a $c^1_t$ exists with a unique unmapped vertex, choose I, Set $p_t = i$
7. CHOOSE some r such that $c^1_{it} = c^2_{rt}$ and for which there has been no mapping yet chosen for $v^2_r$
8. IF such an r exists, GO TO 10
9. Decrement t If $t < 0$, conclude that the graphs have no isomorphism; otherwise, go to 7
10. COMPOSE $C^1_t$ with row t of $D^1$ to yield field $C^1$ Compose $C^2_t$ with row r of $D^2$ to yield $C^{2.}$ GENERATE $C^2$ and $C^2$ by substituting a unique integer for each unique term of $C^1$ and $C^{2.}$ COMPOSE $C^1$ with column i of $D^1$ to yield $C^{1.}$ Compose $C^2$ with column r of $D^2$ to yield $C^{2.}$ Generate $C^1_{t+1}$ and $C^2_{t+1}$ by substituting a unique integer for each unique term of $C^1$ and $C^2$
11. COMPUTE the class count vectors $K^1_{t+1}, K^2_{t+1}$
12. IF $K^1_{t+1} \neq K^2_{t+1},$ go to 7
13. INCREMENT t
14. GO TO 3
END
```

(4)

The algorithm above is an excerpt from [28].

Canonical Graph Labelling Algorithm: This algorithm, which is being implemented as nauty (No AUTomorphisms, Yes?) [29], is known to have exponential running time on some inputs and excellent performance (an implementation that can successfully handle many graphs with a thousand or more vertices within a short time). The canonical graph labelling algorithm was designed for canonically labelling a vertex-coloured graph that locates the generators for the automorphism group. Its entire process was summarised by [30] as follows:

Two graphs *G* and *H* with vertex set [n] are isomorphic if a permutation $\gamma\ \varepsilon\ \Sigma_n$ exists such that $H = G^\gamma$.
Similarly, if Γ acts on X, then it also acts on sequences $(x_i)^k{}_{i=1}$ from *X* using

$$((x_i)^k{}_1)^{\gamma} = (x_i)^k{}_1. \quad (5)$$

The algorithm above is an excerpt from [29]

The algorithm is known to locate the automorphic, as well as the isomorphic, graph.

Corneil and Gotlieb: The procedure of this algorithm to determine if two graphs are isomorphic involves two graphs [31] (i.e. representative and reordered), and check for some criteria. Typically, the representative graph forms the basic condition for determining the isomorphism; for example, if the representative graphs are identical, then, the given graphs will be isomorphic. The algorithm was originally presented in a flowchart form. However, here, it is in text version.

```
START
    Step 1:  CALCULATE $G_R{}^1$ and $G_R{}^2$
    Step 2:  IF $G_R{}^1 = G_R{}^2$
THEN got to Step 3
ELSE
OUTPUT $G_1 \neq G_2$
    Step 3:  CALCULATE $G_r{}^1$ and $G_r{}^2$
    Step 4:  IF $G_r{}^1 = G_r{}^2$
THEN $G_1 = G_2$
ELSE
OUTPUT $G_1 = G_2$ form a counter–example to the conjecture
    END
```

(6)

The algorithm above is an excerpt from [31].

3. Application

Because graphs are used for the representation of objects and their relations, they are extensively used in various areas for visual representation of data. A comparative study of application of the isomorphism algorithm for the matching of graph structures in different areas of research was conducted by [32]. These areas include, but are not limited to, image processing, protein structure and social networks, to determine the structure pattern and matching structures. The use case application of the graph isomorphism is briefly discussed as follows:

Biomedical Use Case: The isomorphic graph detection is applied for chemical bond, as well as protein, structure. In the chemical bond structure application, bond structures are often first converted into graph format and then graph isomorphism is applied to locate the similar bond structure. Furthermore, graph isomorphism is applied for protein structure by matching a given food protein structure with any of the provided food protein structures. In this context, the nodes of the graphs represent the protein, whereas the connection represents their interaction.

Social Network Use Case: Social network, including criminal network, represents actors (people) and their ties (connections). The graph isomorphism is applied for pattern matching (i.e. for detecting positions and location of actors). In an online social network (e.g. Twitter or Facebook), graph isomorphism remarkably aid in evaluating the shortest path between two persons by considering their relationship [33].

Image Processing Use Case: The graph model has long been used for image processing and analysis processes, isomorphism and automorphism are among other analysis techniques employed during image processing. Given that images can be 2D, 3D or 4D objects, determining the isomorphism of two different images requires converting the image in a graph structure and then image matching are implemented (i.e. checking for structure similarity) by using the graph isomorphism algorithm [34].

Computer Design System: In this domain, graph isomorphism is functionally used to ascertain if two states are symmetric or not. Fields such as software engineering, database engineering and computer network see to the usage of isomorphism when carrying out designs [32]

4. Discussion

Graphs are mathematical models used for visual representation of data. As discussed above, node and connections are the two components of a graph, which has different types. One of the structural similarities of two or more (sub)graphs is known as graph isomorphism. Therefore, there are algorithms employed to verify the similarities discussed previously, as well as the areas where this is applicable.

Table 4 critically analyses and summarises the algorithms discussed.

Table 4: Isomorphism algorithms and their respective features.

S/N	Algorithm	Search method	Applied	Pros	Cons
1	Ullman	Refined brute-force enumeration	Used for determining the connectedness of graphs	Copes with undirected subgraph isomorphism	Costly and time consuming when dealing with large graphs
2	VF	Depth-first strategy and tree pruning	Knowledge extraction from large chemical database, retrieval of solid models from database	Less time consuming when matching large number of nodes compared with the Ullman algorithm	Has higher computational complexity than the Ullman algorithm
3	VF2	Depth-first strategy and tree pruning	Image processing/matching	Matching time independent on the number of nodes	Same as VF
4	Canonical graph labelling	Graph transformation into canonical form	Test for colour-preserving isomorphisms of vertex-coloured graphs	Ability to prune search tree by using an indicator function	Record an exponential running time while working on some graphs
5	Schmidt and Druffel	Backtracking	Used in fields such as information retrieval and network theory	Powerful for processing large class of graphs	Not guaranteed to run in polynomial time
6	Corneil and Gotlieb	Uses power function of a graph (i.e. the order of the given graph)	Locates a given chemical compound (i.e. chemistry application)	Efficient on finite graphs, based on conjecture	Inefficient for graphs whose strong regularity is a function of n
7	MLC	Merging Sort	For matching unique node labelled graph	Lexicographic ordering for speeding up list sorting, matches sorted adjacency lists by using the form of merge sorting	Unable to work on graphs with duplicate nodes

The survey paper on isomorphic algorithms conducted by [35] reviewed a total of five algorithms used for this purpose. However, this paper reviewed the newly developed algorithms, briefly detailed graph and its type, graph analysis and its basic theory, isomorphic algorithms (old and new) and also application use cases.

5. Conclusion

A graph analysis focuses on investigating any data representation as vertices and edges not only to visualise but also locate and extract hidden insights from such representation. An isomorphic graph was intensively reviewed in this research. Thus, the algorithm used in this type of graph analysis (both old and recent ones) was reviewed, each algorithm was stated, pros and cons were discussed and various fields of application were identified. The result showed that the (sub)graph isomorphism is a useful technique for graph analysis and is applicable to numerous real-life cases. This research highlighted the need, use and importance of the sub(graph) isomorphism in detail.

References

[1] Pupazan, E. (2011). Social networking analytics. BMI Paper, VU University Amsterdam, Social Networking, Vol.7 No.3, July 18, 2018

[2] Garey, M. R. (1979). A Guide to the Theory of NP-Completeness. Computers and Intractability Book, Volume 1, 1st Edition.

[3] Xu, J., & Chen, H. (2005). Criminal network analysis and visualization. Communications of the ACM, 48(6), 100-107

[4] Sparrow, M. K. (1991). The application of network analysis to criminal intelligence: An assessment of the prospects. Social networks, 13(3), 251-274.

[5] R. Fox, “Criminal Network and Link Prediction,” vol. 5, no. 1976, pp. 265-288, 2010.

[6] Hutchins, C. E., & Benham-Hutchins, M. (2010). Hiding in plain sight: criminal network analysis. Computational and Mathematical Organization Theory, 16(1), 89-111.

[7] Hopcroft, J. E. & Wong, J. K. (1974). Linear time algorithm for isomorphism of planar graphs (preliminary report). In Proceedings of the sixth annual ACM symposium on Theory of computing (pp. 172-184). ACM.

[8] M. De Santo, P. Foggia, C. Sansone, and M. Vento, “A large database of graphs and its use for benchmarking graph isomorphism algorithms,” Pattern Recognit. Lett., vol. 24, no. 8, pp. 1067-1079, 2003n

[9] Radford, L. (2010). Layers of generality and types of generalization in pattern activities, PNA, 4(2), 37-62.

[10] Wasserman, W. W., & Sandelin, A. (2004). Applied bioinformatics for the identification of regulatory elements. Nature Reviews Genetics, 5(4), 276.

[11] Miller, M. (2011). An overview of the degree/diameter problem for directed, undirected and mixed graphs. In Proceedings of the 3rd International Workshop on Optimal Networks Topologies IWONT 2010 (pp. 335-345). Iniciativa Digital Politècnica.

[12] Oh, K. K., Park, M. C., & Ahn, H. S. (2015). A survey of multi-agent formation control. Automatica, 53, 424-440.

[13] Gao, X., Xiao, B., Tao, D., & Li, X. (2010). A survey of graph edit distance. Pattern Analysis and applications, 13(1), 113-129.

[14] Gardner, R., Bobga, B., Nguyen, C., & Coker, G. (2005, January). Some Graph, Digraph, and Mixed Graph Results Concerning Decompositions, Packings, and Coverings. In Abstract Presented at Joint Meeting of the AMS and the MAA, Atlanta GA, Abstract (No. 1003-05, p. 120).

[15] Mohammed, A. M. (2017). Mixed Graph Representation and Mixed Graph Isomorphism. Gazi University Journal of Science, 30(1), 303-310.

[16] Gross, J. L., & Yellen, J. (2005). Graph theory and its applications. Chapman and Hall/CRC.

[17] Fan, W. (2012). Graph pattern matching revised for social network analysis. In Proceedings of the 15th International Conference on Database Theory (pp. 8-21). ACM.

[18] Haxhimusa, Y. (2007). The structurally Optimal Dual Graph Pyramid and its application in image partitioning (Vol. 308). IOS Press.

[19] Adopted from: Hauskrecht, Milos. 2012. “CS 441 Discrete Mathematics” for CS Lecture 25. Link: https://people.cs.pitt.edu/~milos/courses/cs441/ Accessed on: 12/9/2018

[20] Shafie, T. (2015). A Multigraph Approach to Social Network Analysis. Journal of Social Structure, 16.

[21] Sarvari, H., Abozinadah, E., Mbaziira, A., & Mccoy, D. (2014, May). Constructing and analyzing criminal networks. In Security and Privacy Workshops (SPW), 2014 IEEE (pp. 84-91). IEEE

[22] Levchenko, Kirill et al. (2011). "Click trajectories: End-to-end analysis of the spam value chain.” In 2011 IEEE symposium on security and privacy, pp. 431-446. IEEE.

[23] McCoy, D et al. (2012). Priceless: The role of payments in abuse-advertised goods. In Proceedings of the 2012 ACM conference on Computer and communications security (pp. 845-856). ACM.

[24] J. R. Ullmann, (1976). “An Algorithm for Subgraph Isomorphism,” J. ACM, vol. 23, no. 1, pp. 31–42.

[25] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, (1999). “Performance evaluation of the VF graph matching algorithm,” Proc.-Int. Conf. Image Anal. Process. ICIAP 1999, no. February, pp. 1172-1177.

[26] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, (2004). “A (sub)graph isomorphism algorithm for matching large graphs,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 10, pp. 1367–1372.

[27] L. Fu and L. Chandra, (2012). “Optimized Backtracking for Subgraph Isomorphism,” Int. J. Database Manag. Syst., vol. 4, no. 6, pp. 1–10.

[28] Schmidt, D. C., & Druffel, L. E. (1976). A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. Journal of the ACM (JACM), 23(3), 433-445.

[29] B. D. McKay, (1981). “Practical Graph Isomorphism,” vol. 30.

[30] S. G. Hartke and A. J. Radcliffe, (2007). “McKay’s Canonical Graph Labeling Algorithm,” Contemp. Math., vol. 0000, pp. 99-112

[31] D. G. Corneil and C. C. Gotlieb, (1970). “An Efficient Algorithm for Graph Isomorphism,” J. ACM, vol. 17, no. 1, pp. 51-64
[32] Somkunwar, R., & Vaze, V. M. (2017). A Comparative Study of Graph Isomorphism Applications. International Journal of Computer Applications, 162(7).
[33] Fan, W. (2012). Graph pattern matching revised for social network analysis. In Proceedings of the 15th International Conference on Database Theory (pp. 8-21). ACM.
[34] Lézoray, O., & Grady, L. (2012). Graph theory concepts and definitions used in image processing and analysis. Image processing and analysing with graphs: theory and practice. CRC Press, Boca Raton, FL, 1-24.
[35] Voss, S., & Subhlok, J. (2009). Performance of general graph isomorphism algorithms (Doctoral dissertation, Coe College).