An Efficient Alternative to Subgraph Isomorphism and Its Advantages

Wenjie Zhang
Tsinghua University
Shenzhen, China
zhangwj19@mails.tsinghua.edu.cn

George P. Chan
Loudonville Christian School
Loudonville, NY
gchan@lcscourses.net

Wai Kin (Victor) Chan*
Tsinghua University
Shenzhen, China
chanw@sz.tsinghua.edu.cn

ABSTRACT

Subgraph Isomorphism is a fundamental problem in graph theory. It has many applications in social network analysis, molecular investigations, knowledge graphs, etc. Given a Query Graph and a Data Graph, the target of Subgraph Isomorphism, i.e., Subgraph Matching, is to determine if this Query Graph is isomorphic to any subgraph of the Data Graph. This work proposes a new type of Query Graph, combined with multiple general Query Graphs. We call it Compulsory-Optional Query Graph (CO Query Graph). This new type of Query Graph contains all the vertices in the combined general Query Graph, and each vertex corresponds to a search priority. Based on CO Query Graph, the previous multiple match processes can be reduced to one. It tremendously improves search efficiency. The Subgraph Isomorphism based on this new kind of Query Graph is an extension and improvement of the previous Subgraph Isomorphism studies. We propose a backtracking-pruning-based CO solver (BPC). This algorithm builds on the backtracking-pruning framework. BPC modifies the output criterion and matching conditions to satisfy the CO query context. A case study of real-world graph data illustrates that BPC built on CO Query Graph is more efficient than conventional Query Graphs. To verify the effectiveness of our method, we conducted experiments on the synthetic graph and real-world data. The results show that the BPC can significantly reduce the search space and improve the search efficiency in the recursive calls and the response time. Experiments resulting from synthetic graph data analysis allow us to primarily identify the critical factor that affects the efficiency of the BPC primarily.

CCS CONCEPTS

• **Information systems** → **Network data models**.

KEYWORDS

subgraph isomorphism, graph retrieval, subgraph matching

ACM Reference Format:
Wenjie Zhang, George P. Chan, and Wai Kin (Victor) Chan*. 2022. An Efficient Alternative to Subgraph Isomorphism and Its Advantages. In *2022 5th International Conference on Algorithms, Computing and Artificial Intelligence (ACAI 2022), December 23–25, 2022, Sanya, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3579654.3579768

ACAI 2022, December 23–25, 2022, Sanya, China

ACM ISBN 978-1-4503-9833-6/22/12.
https://doi.org/10.1145/3579654.3579768

1 INTRODUCTION

The goal of information retrieval is to obtain the required information from a large amount of informative data, which is typically those informative data are collected in a database. We construct the information needed into a formal statement and regard it as a query. With the development of computer technology, the size of a database has become larger and larger. Thus, information retrieval requires a sophisticated design to handle large-scale data.

Structural data are prevalent objects in information retrieval studies. The graph is one type of Structural data. Due to its convenience of modeling and processing, many retrieval systems seek the help of graphic data. Subgraph matching, a fundamental problem in graph theory, plays an essential role in graphic information retrieval. Given a graphic pattern, it will search for the most relevant data and give it back to the users. It plays a crucial role in many disciplines, including social network [2, 3], biochemistry [12], natural language processing [9], and computer version[4]. In subgraph matching, the queries are in graphic manner. We call them query graphs. Similarly, the information retrieval process gives a collection of graphical information contained in a database. We view one piece of it as the data graph.

Subgraph matching gives a more organized, efficient, and diversified way to retrieve information from a graph perspective. This issue is designed to answer the following questions: (1) Does this query graph exist in a given data graph? (2) How many subgraphs in the data graph [11] are isomorphic to this query graph? (3) How does the query graph embed those isomorphic subgraphs in relevant data graphs?

Subgraph Isomorphism has been proven as an NP-Complete problem. It is easy to verify whether a given solution is correct or not, whereas it is difficult to figure out the correct solutions. That means with the growth of vertices number, solving this problem has become time-consuming [7]. Not only that, its matching conditions are too strict to miss a lot of relevant, valuable data graphs that are partially matched with its query graph. Using a subgraph isomorphism solver is not an optimized way to handle subgraph matching.

We introduce CO matching designs a new type of query graph with three search priorities to lose the matching constraints. We use three kinds of vertices to construct CO query graph, i.e., compulsory and optional vertices. Compulsory vertices contain the ultimate information that the solver must find. Optional vertices are the unique vertices that can not be shown in the data graph. Not only

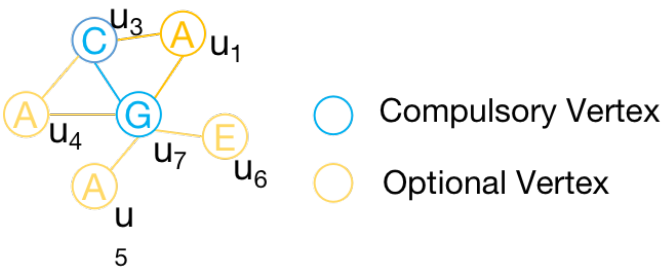

Figure 1: An example of CO query graph

can it precisely match the relevant data graphs, but also can improve the computational efficiency compared to the strict query graphs used by subgraph isomorphism.

To tackle the CO matching problem, we propose a backtracking-pruning-based algorithm, a backtracking-pruning-based CO solver (BPC). This algorithm is specifically designed by changing the output criterion and matching conditions. We conduct experiments on three synthetic and two real-world graph datasets to evaluate the performance of BPC. Results show that it has achieved great improvements on search efficiency.

Main contributions of this study:

(1) In this work, we follow a subject of CO matching and propose BPC to solve this subgraph matching problem with CO query graph.
(2) We conduct systematic experiments to evaluate the advantages of CO query graphs over conventional query graphs used in Subgraph Isomorphism.
(3) Our work alleviates the burden of conventional querying work and accelerates the graph information retrieval process by matching label first.

2 PROBLEM DEFINITION AND PRELIMINARY

Besides Maximum Common Subgraph, there are various subgraph matching variants. Yves Deville [16] proposed approximating subgraph matching. It divided the vertices of its query graph into two classes, i.e., the compulsory and the optional vertices. However, their work builds on VF algorithm which has been developed far more efficient. While Hoffmann [8] gives an alternative matching problem between Subgraph Isomorphism and Maximum Common Subgraph, named k-less matching. It aims to match a query graph with all but k vertices to the data graph. Moreover, the maximum common partial subgraph problem [13], which concentrates on maximizing the mapping edges, can also serve as a tool for graph retrieval.

Figure 1 gives an example of CO query graph Q in which compulsory vertices are in blue; and optional vertices are in yellow.

We classify the vertices of CO query graph into three types: the Compulsory vertex $V_{compulsory}$ and the Optional vertex $V_{optional}$. $\mathcal{M}$ is a mapping function from vertices in Q to that in D. If a data graph D is matched with query graph Q, the vertices with a compulsory label of query graph Q must be contained in the data graph D. The vertices with optional labels should be included as much as possible in the data graph D for full information extraction.

In this thesis, we mainly consider undirected, vertex-labeled graphs, D=(V, E, L), where V is the set of vertices, E is the set of edges, and f is the mapping function from vertices to their labels. All graphs in this work are connected. We use the adjacency matrix to store the graphs. From this perspective, we can handle the CO matching problem by changing the output criterion and matching conditions. The vertices order, manifesting as the diagonal of an adjacency matrix of data graph D, is regarded as the built-in order of vertices in Q. We denote the indexed vertex set $u1, u2, ..., u|V|$.

3 BACKTRACKING-PRUNING-BASED CO MATCHING SOLVER

3.1 Filtering Process

Firstly, we implement a simple filtering algorithm to drop out some unrelated data graph vertices in advance. It filters the vertices of data graph with label not showing in the query graph.

Given the label functions of query graph and original Data Graph $f_Q : V_Q \leftarrow L_{V_Q}, f_O : V_O \leftarrow L_{V_O}, L_{V_O}$ and L_{V_O} are the label sets of data graph and query graph, respectively. The filtering process will get data graphs with $f_D : V_D \leftarrow L_{V_D}$, where $set(L_{V_D}) = set(L_{V_Q})$.

We take the neat data graphs D, whose label set is equal to that of query graph Q, as the data graph to be matched.

3.2 Backtracking-pruning-based CO Solver (BPC)

We invent a novel algorithm based on a backtracking-pruning framework to settle CO matching. We call it **B**acktracking-**P**runing-based **C**OF solver (BPC). To settle CO matching, the main challenges compared to the basic backtracking algorithm are listed as follows:

(1) Problem 1: The vertex order in the query graph generally differs from its corresponding order organized in the data graph.
(2) Problem 2: The duplicated vertices, which have the same label and same relationships with other vertices in graph, will result in several mapping solutions.
(3) Problem 3: In case the data graph has fewer vertices compared to the query graph, our method should find the intersection of the query and data graph vertices.
(4) Problem 4: How to distinguish the current vertex type as the primary concern of BPC.

We present the outlines of BPC in algorithm 1.

Unlike the general subgraph matching way, CO matching is simplified to match the labels of query graph Q with that of data graph D instead of their vertex indexes. This makes the search process mainly focus on the label matching instead of the vertex index. If there is a one-to-one relationship between the vertex label and index, i.e., one label gets one vertex index, it just needs to return the vertex's index. If there are one-to-many relationships, i.e., one label gets several corresponding vertices, the search tree will branch to ensure that the results cover all correct solutions. That settles problem 2. Technically, it needs a decision tree model. We illustrate this process in Figure 2.

At the beginning of BPC, the first vertex in the data graph D is set to match the vertex with the same label in the query graph Q. Then, the algorithm visits the next vertex u in the query graph Q to query

Algorithm 1 Backtracking-pruning-based CO Solver (BPC)

Require: Data graph$D$, query graph Q, mapping solution $\mathcal{M} = \emptyset$
Ensure: Mapping solution $\mathcal{M}$
  if $\mathcal{M}(u)$ satisfied the output criterion **then**
    Output($\mathcal{M}(u)$)
  else
    if still have vertex in D not been visited **then**
      Visit the next v in D
      BPC(data graph$D$, query graph Q, $\mathcal{M}$)
    end if
    if candidate $v = C(u)$ and u satisfied the matching conditions **then**
      Include (u, v) into $\mathcal{M}$; mark u queried
      Query next u in Q
      BPC(data graph$D$, query graph Q, $\mathcal{M}$)
      Mark u un-queried; delete (u, v) from M
    end if
  end if

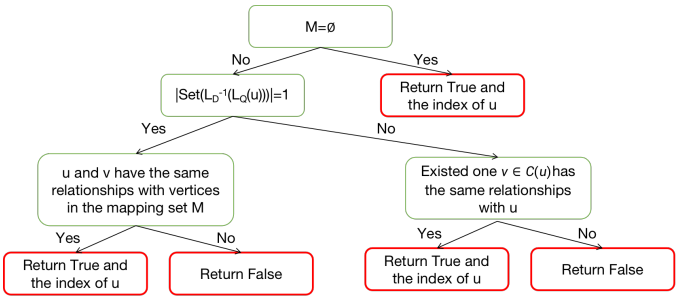

Figure 2: The decision tree of label match process

if any unmatched vertex in data graph D can match it. (The match conditions are elaborated above.) If v and u match, (u, v) will belong to $\mathcal{M}$. The search process will partition into two search branches when it finds the current matched vertex pair. One is included in the matched vertex pair (u, v). Another is to skip this matched vertex pair (u, v) and keeps the current vertex u unmatched to find the candidate $C(u)$ of another vertex in data graph D. If u and v are not matched, BPC will visit the next vertex in data graph D, namely the next element in the diagonal of A. If it does not meet the output criterion, the algorithm will return to the branch point to search for another path. Only by broadening the search scope through this way can we meet all the possible solutions and solve problem 3.

The match conditions of the two vertices u and v matching are outlined as follows:

(1) The labels of the current vertex pair match each other, i.e., $f_D(v) = f_Q(u)$.
(2) There still have unvisited vertices in the data graph.
(3) The number of vertices remaining not being visited in data graph D is more than that of the compulsory vertices in query graph, namely $\|V(D)\| - \|V_{visited}(D)\| \geq$ unmathed$\|V_{compulsory}\|$, where is the number of un-queried compulsory vertices.
(4) The adjacent vertices of $v \in C(u)$ is identified with that of u. That normally uses the adjacency matrix to compare.

Table 1: Real-world dataset statistics

Name	Num.Graphs	Avg.Vertices	Avg.Edges
BZR_MD	306	21.30	225.06
MSRC_9	221	40.58	97.94

(5) The number of adjacent vertices,i.e., the degree, of u is no more than that of $v \in C(u)$, namely $\|d(u)\| \leq \|d(v)\|$, where $d(u)$ is the degree of u[6].

If all match conditions above are qualified, the candidate vertex $C(u) = v$ in data graph can be included in $\mathcal{M}$.

To improve the search efficiency, we also provide the output criterion that can stop the unpromising search work earlier. Here are the criteria, i.e., output criterion, to end the recursions:

(1) All vertices in the data graph D have been visited.
(2) As many as possible optional vertices in the query graph have been mapped.
(3) All compulsory vertices have been included in the candidate sets $C(u)$.

This solver outputs the final result if all output criterion above are qualified. That helps us figure out problem 4.

4 EXPERIMENTS AND ANALYSIS

4.1 Experiments Setting

4.1.1. Synthetic Datasets

There are three graph datasets generated by three different graph synthetic algorithms, i.e., Random Networks generated by Erdos-Renyi (ER) [5] algorithm and Scale-Free Networks generated by Barabasi-Alber (BA) [1]. To make a fair comparison, we let each data graph contain 20, 30 and 50 vertices. All query graphs have 6 vertices. We control the generating factors of each random synthetic methods to see its impact on the efficiency of BPC. Each generating graph is randomly created ten times. We take the mean value of them to produce the final results.

Also, we do the ablation study on strict query graphs. All CO query graphs are attached to the tag [0,0,1,1,1,1]. , so that CO query graph setting will result in $C_3^2 = 3$ strict query graphs. We install our BPC and a basic backtracking algorithm to the CO query graphs and the strict query graphs, respectively.

4.1.2. Real-world Datasets

We conduct experiments on BZR_MD [10, 15], a small molecules graph dataset, and MSRC_9 [14], a computer vision graph dataset to evaluate the performance of BPC on the real-world datasets. Table 1 gives the statistics of those two real-world graph datasets. Both of them are vertex-labeled, undirected graphs.

4.1.3. Implement Details and Evaluated Method

This work is implemented on Intel(R) Core(TM) i7-9800X CPU at 3.80GHz. All experiments are conducted on the Windows system. We use python as the programming language.

4.2 Real-world Case Study

We chose a real-world case from the dataset generated from the semantic spinning tree. The input of this example is illustrated in

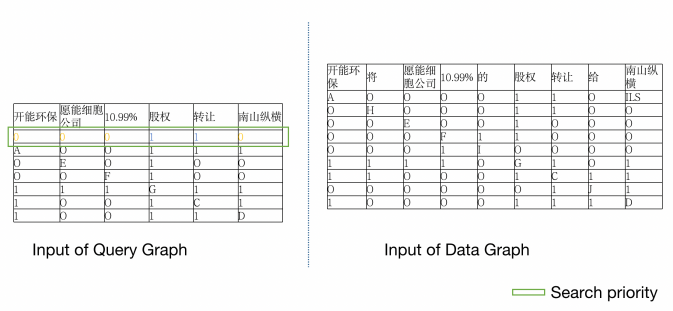

Figure 3: Example of real-world dataset

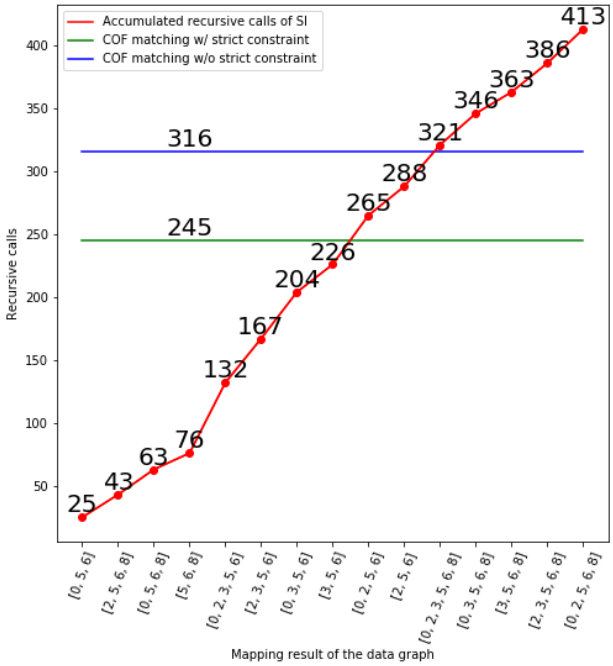

Figure 4: Recursive calls comparison between strict query graph and the CO query graph

Figure 3. The vertex is the word segmentation of a sentence. The diagonal is the label of each vertex, and the element of **1** in upper and lower triangular matrix implies the connection between two vertices. The first raw is the search priorities label of those vertices, where **0** denotes optional vertex and **1** denotes compulsory vertex. We conduct our real-case study on this example.

4.2.1. The Effect of Alleviating the Burden of Query Graph Building Work

As illustrated by Figure 3, we only get two compulsory vertices in the query graph. If we want to get all the results of the diverse strict query graphs, we need to build $C_5^2 = 15$ query graphs. We implement VF2 for all those strict query graphs. The red line in Figure 4 denotes the accumulating recursive calls of those 15 cases. The blue line is the up-bound of the recursive calls of the CO query graph. As we can see, with only one CO query graph, we can get all the results with only 316 recursive calls at most. A strict constraint will result in 245 recursive calls. However, there will be a total requirement of 413 recursive calls through the strict query graph to get all the results we need. This figure indicates that our CO query graph not only alleviates the burden of the query graph building work but also accelerates the search process.

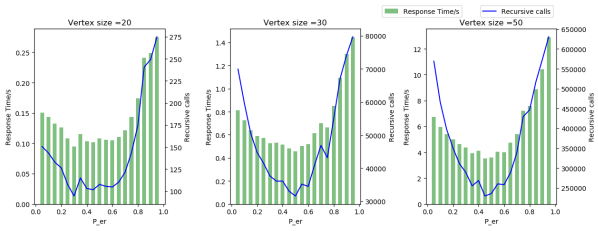

Figure 5: Performance of BPC on random graph

4.3 Synthetic Dataset

4.3.1. Random Networks

ER random graph is a network in which the vertex pair connect to each other with probability P_{er}. We choose two vertices in the query graph as the compulsory vertex. Figure 5 shows the recursive calls and the response time changing along with the probability of the connection between vertex pair. It demonstrates that it is extremely hard to find out the result when the vertices connect to each other with a small probability or with a large probability. When the vertices connect to each other with the probability of 0.5, the recursive calls and the response time are at their lowest value. When the vertices connect to each other with the probability of 0.95, the recursive calls and the response time are at their highest value. As we can see, the density of edges affects the search efficiency mostly when it is extremely large. This phenomenon has since been discovered to apply to all ER random networks with 20,30,50 vertices.

Figure 6 shows the difference in recursive calls between CO matching and strict Subgraph Isomorphism matching. It changes along with the probability of vertices connection. For the ER networks with 50 vertices, the CO matching shows superiority over the strict Subgraph Isomorphism for graphs with higher and lower vertices connection probabilities, such as probabilities below 0.25 and beyond 0.7. Additionally, they perform similarly over vertices with connection probabilities ranging from 0.3 to 0.65. Our CO query graph requires significant work to obtain the mapping solution only on the graph with a 0.6 vertices connection probability. The more vertices the data graphs contain, the more recursive calls will be saved by CO matching.

4.3.2. Scale-Free Networks

We study the scale-free networks with 20 vertices. The recursive call and the response time are given in Figure 7. This figure shows that the recursive calls decrease with the growth of connected edges. The scale-free graph case will have the best search efficiency when the number of inserted edges in the Data Graph is close to the number of vertices in the query graph. This phenomenon is applied to all scale-free networks with 20, 30, and 50 vertices.

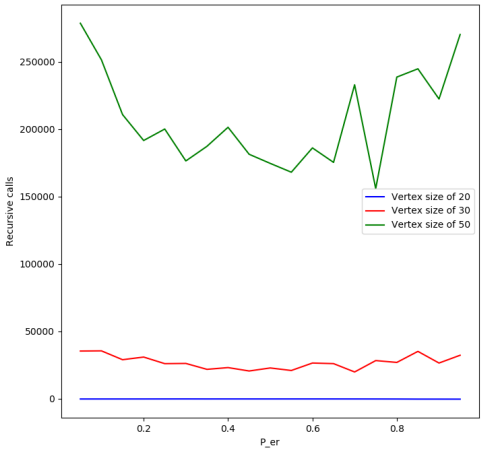

Figure 6: Difference between CO matching and strict SI matching on random graphs

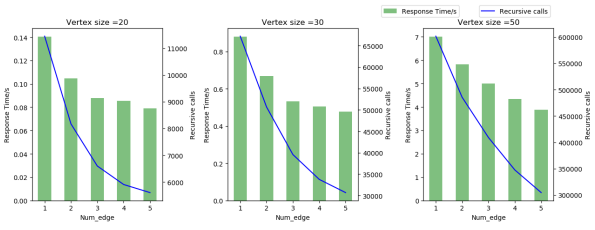

Figure 7: Performance of BPC on scale-free graphs

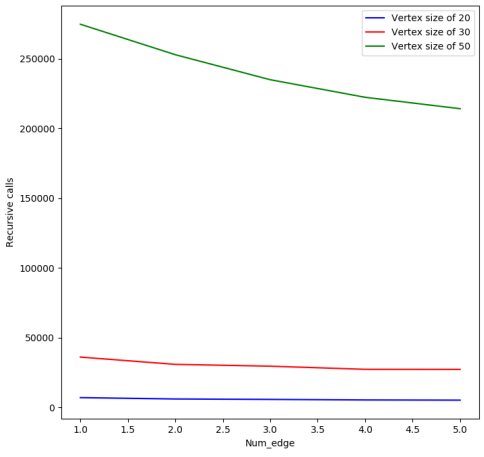

Figure 8: Difference between CO matching and strict SI matching on scale-free graphs

Table 2: Recursive calls: CO matching vs. strict SI matching

Name	Avg.CO recursive calls	Avg.strict SI recursive calls	Avg.Difference recursive calls
BZR_MD	26461.51	38750.36	12288.85
MSRC_9	181205.24	315866.29	134661.06

Table 3: Response time: CO matching vs. strict SI matching

Name	Avg.CO response time	Avg.strict SI response time	Avg.Difference response time
BZR_MD	643.80	378.31	-265.48
MSRC_9	2957.30	1970.81	-986.49

Figure 8 gives the difference between CO matching and Strict Subgraph Isomorphism matching on small-world graph data. As we can see, the CO query format significantly increases the effectiveness of the search with the growth of graph vertices. Scale-free data graph with 20 vertices has little advantage over graphs with larger vertices. The search format of the CO query graph significantly reduces the number of recursive calls regardless of the number of inserted edges. The results show that the CO query graph format is a more useful query format for scale-free networks subgraph matching.

4.4 Real-world Dataset

We implement BPC on CO matching and a basic backtracking algorithm on strict SI matching on the two real-world dataset. Table 2 gives the results of the recursive calls on two real-world datasets. Table 3 shows the response time results. We give the equation to calculate the difference between CO matching and strict SI matching as shown in 1, where S denotes the result of recursive calls or response time of strict SI matching, N.CO denotes that of CO matching and i denotes i_{th} data graph. This equation is suitable for both difference calculation of recursive calls and response time.

$$\text{Avg.diff} = \frac{\sum_{i=1}^{n} \text{N.S}_i \text{ - N.CO}_i}{n} \tag{1}$$

In conclusion, BPC has better performance in not only alleviating the burden of query graph building work but also the recursive calls. However, it will spend more time finding the correct solutions.

5 CONCLUSION

In this work, we developed an algorithm BPC to solve the CO matching. By classifying the vertices of CO query graph Q into compulsory and optional vertex, multiple search work from database can use only one CO query graph. Hence the heavy work of query graph building will be alleviated. A real-world case verifies this result.

We also examined the effectiveness of BPC on various synthetic graphs. The results indicate that the ER random graph is primarily influenced by the vertices' connection probability. The number of neighborhoods primarily affects the small-world graph. The scale-free graph is primarily impacted by the inserted edge number

proportions. We also conducted experiments on two real-world graph datasets. Results show that our method can accelerate the search process and find the result as accurately as possible. In addition, this method creatively converts the subgraph matching into label matching which provides a novel approach to dealing with the classic order problem.

ACKNOWLEDGMENTS

This research was funded by the Shenzhen Science and Technology Innovation Commission (JCYJ20210324135011030), Science and Technology Innovation Committee of Shenzhen-Platform and Carrier (International Science and Technology Information Center), High-end Foreign Expert Talent Introduction Plan (G2021032022L), Guangdong Pearl River Plan (2019QN01X890), and National Natural Science Foundation of China (Grant No. 71971127).

REFERENCES

[1] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science* 286, 5439 (1999), 509–512.
[2] Wai Kin Victor Chan. 2017. Agent-based and regression models of social influence. In *2017 Winter Simulation Conference (WSC)*. IEEE, 1395–1406.
[3] Wai Kin Victor Chan and Cheng Hsu. 2015. When human networks collide: the degree distributions of hyper-networks. *IIE Transactions* 47, 9 (2015), 929–942.
[4] Pedro F Felzenszwalb and Ramin Zabih. 2010. Dynamic programming and graph algorithms in computer vision. *IEEE transactions on pattern analysis and machine intelligence* 33, 4 (2010), 721–740.
[5] Edgar N Gilbert. 1959. Random graphs. *The Annals of Mathematical Statistics* 30, 4 (1959), 1141–1144.
[6] Tae Wook Ha, Jung Hyuk Seo, and Myoung Ho Kim. 2018. Efficient searching of subhypergraph isomorphism in hypergraph databases. In *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 739–742.
[7] Juris Hartmanis. 1982. Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Review* 24, 1 (1982), 90.
[8] Ruth Hoffmann, Ciaran McCreesh, and Craig Reilly. 2017. Between subgraph isomorphism and maximum common subgraph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.
[9] Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. 2017. Answering natural language questions by subgraph matching over knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering* 30, 5 (2017), 824–837.
[10] Nils Kriege and Petra Mutzel. 2012. Subgraph matching kernels for attributed graphs. *arXiv preprint arXiv:1206.6483* (2012).
[11] Xin Liu, Haojie Pan, Mutian He, Yangqiu Song, Xin Jiang, and Lifeng Shang. 2020. Neural subgraph isomorphism counting. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1959–1969.
[12] Oliver Mason and Mark Verwoerd. 2007. Graph theory and networks in biology. *IET systems biology* 1, 2 (2007), 89–119.
[13] Samba Ndojh Ndiaye and Christine Solnon. 2011. CP models for maximum common subgraph problems. In *International Conference on Principles and Practice of Constraint Programming*. Springer, 637–644.
[14] Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. 2016. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning* 102, 2 (2016), 209–245.
[15] Jeffrey J Sutherland, Lee A O'brien, and Donald F Weaver. 2003. Spline-fitting with a genetic algorithm: A method for developing classification structure- activity relationships. *Journal of chemical information and computer sciences* 43, 6 (2003), 1906–1915.
[16] Stéphane Zampelli, Yves Deville, and Pierre Dupont. 2005. Approximate constrained subgraph matching. In *International Conference on Principles and Practice of Constraint Programming*. Springer, 832–836.