Graph Association Analyses for Early Drug Discovery

Wenfei Fan*
Shenzhen Institute of Computing Sciences, China
University of Edinburgh, UK
Beihang University, China
wenfei@inf.ed.ac.uk

Daji Li
Peiyu Liang
Shenzhen Institute of Computing Sciences, China
lidaji@sics.ac.cn
liangpeiyu@sics.ac.cn

Shuhao Liu
Yaoshu Wang
Shenzhen Institute of Computing Sciences, China
shuhao@sics.ac.cn
yaoshuw@sics.ac.cn

Yiming Wang
Shenzhen Institute of Computing Sciences, China
wangyiming@sics.ac.cn

Min Xie
Shenzhen Institute of Computing Sciences, China
xiemin@sics.ac.cn

Runjie Zhang
Shenzhen Institute of Computing Sciences, China
zhangrunjie@sics.ac.cn

ABSTRACT

We demonstrate MedHunter, a system for assisting the early stage of drug development. MedHunter builds a biomedical knowledge graph DDKG by integrating data from eleven biochemical libraries and data banks, and aligning entities from different data sources by means of heterogeneous entity resolution. It identifies drug-disease associations and protein-protein interactions in DDKG by employing graph association rules (GARs). GARs use graph patterns to extract relevant entities and embed ML models as predicates. MedHunter discovers GARs from DDKG and incrementally enriches DDKG with external data; it cleans DDKG with a special form of GARs. We demonstrate MedHunter for its (a) interfaces, (b) data enrichment/cleaning, and (c) applications in target identification, drug-drug interaction and protein-protein interaction.

PVLDB Reference Format:
Wenfei Fan, Daji Li, Peiyu Liang, Shuhao Liu, Yaoshu Wang, Yiming Wang, Min Xie, and Runjie Zhang. Graph Association Analyses for Early Drug Discovery. PVLDB, 17(12): 4293 - 4296, 2024.
doi:10.14778/3685800.3685858

1 INTRODUCTION

Drug discovery is the process of new drug identification, starting from target selection and validation, through preclinical screening, to clinical trials [13]. It is time-consuming and costly. The development of a new drug takes 10–15 years, costs around 2 billion US dollars, and typically has a high risk of failure (>90%) [22].

To accelerate drug discovery, reduce the cost and increase the success rate, machine learning (ML) models have been explored to identify drug-disease associations (DDAs), drug-drug interactions (DDIs), and protein-protein interactions (PPIs). However, it is costly to train accurate deep-learning models, and ML predictions often have false positives (FPs) and false negatives (FNs). Moreover, ML predictions "still lack reliable explanations that are crucial to drug repurposing and ADR (adverse drug reaction)" [22]. Worse yet, noise is introduced by "unavoidable inaccuracy" of the models, *e.g.,* nonexistent relations and inaccurately named entities.

Can we do better? Is it possible to develop a cost-effective method to reduce false positives and false negatives of ML predictions, and moreover, enrich and clean the data from different data sources?

MedHunter (Section 2). In response to this, we have developed a system to assist early drug discovery, known as MedHunter [1]. MedHunter aims to assist (1) target identification, to identify the right biological molecules or cellular pathways that can be modulated by drugs to achieve therapeutic benefits, in which PPIs are often needed, (b) drug repurposing, to reuse existing drugs for a new disease, and (c) ADR, to disclose undesirable impacts of drugs.

The need for these is evident. As an example, target identification is perhaps the most crucial initial step in drug discovery, and influences the chances of success at every step of drug development. Traditional target identification easily takes from years to decades. MedHunter helps select candidate targets and speed up the process.

Compared to ML models, MedHunter offers the following.

Graph association. MedHunter infers DDAs and DDIs from graphs based on both ML predictions and logic deduction. It employs a class of graph association rules (GARs) of the form $Q[\bar{x}](X \rightarrow p_0)$ [10]. Here Q is a graph pattern to extract related entities and their relationships/interactions; and $X \rightarrow p_0$ is a dependency on the entities. Moreover, ML models may be embedded as predicates for link/node classification. This allows MedHunter to directly leverage existing ML models, and enhance their predictions with logical reasoning.

Accuracy and explanation. MedHunter deduces association p_0 as a logical consequence of GARs with certainty [12], using accumulated ground truth, *i.e.,* if the GARs and ground truth are correct, so is p_0. Moreover, it filters FPs and FNs of ML model $\mathcal{M}$ by adding logic predicates in precondition X. A PPI deduced by MedHunter in May 2022 coincides a finding published in Nature in the same month [17].

In addition, if p_0 is an ML model $\mathcal{M}$ for predicting DDIs, DDAs, PPIs or ADR, we can discover pattern Q and conditions X for rules of the form $Q[\bar{x}](X \rightarrow \mathcal{M})$ to provide rational behind the predictions of $\mathcal{M}$. This is possible for GNN-based models, which are no more expressive than two-variable first-order logic with limited counting [6, 14]. Such interpretability is one of the key advantages of MedHunter. It enables researchers to get a glimpse of the inner

*Author names are listed in alphabetical order.

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685858

workings of $\mathcal{M}$, so as to make improvement with novel insights.

A uniform KG. MedHunter builds DDKG, a drug-disease knowledge graph (KG) in which it discovers GARs and deduces DDAs, DDIs and PPIs. DDKG consists of data from eleven libraries and data banks, *e.g.,* CTD [3] and BioGrid [2]. It aligns entities from different graphs [9], and incrementally enriches DDKG by external data [11].

Biomedical data cleaning. MedHunter cleans DDKG by employing a special form of GARs [7]. It (incrementally) discovers such GARs, detects errors and fixes the errors in DDKG with the GARs.

Demonstration (Section 3). We give a guided tour of MedHunter, from rule discovery, rule deduction of DDAs, DDIs and PPIs, to aligned entities in DDKG. Participants are invited to interact with MedHunter and experience how it helps in DDAs, DDIs and PPIs.

2 AN OVERVIEW OF MEDHUNTER

This sections presents the association rules (Section 2.1), knowledge graph (Section 2.2) and architecture (Section 2.3) of MedHunter.

2.1 Graph Association Deduction

MedHunter employs graph association rules (GARs). Graphs are modeled as $G = (V, E, L, F_A)$, where V is a finite set of vertices; $E \subseteq V \times L(E) \times V$ is a finite set of edges (v, l, v') with label l; each v in V is labeled with its "content" $L(v)$; and each vertex $v \in V$ carries a tuple $F_A(v) = (A_1 = a_1, \ldots, A_n = a_n)$ of *attributes* of a finite arity.

GARs. GARs are defined with a pattern and a dependency. A *graph pattern* is $Q[\bar{x}] = (V_Q, E_Q, L_Q, \mu)$, where (1) V_Q (resp. E_Q) is a finite set of vertices (resp. edges), (2) L_Q assigns a label $L_Q(u)$ (resp. $L_Q(e)$) to each pattern vertex $u \in V_Q$ (resp. edge $e \in E_Q$), (3) $\bar{x}$ is a list of distinct variables, and μ is a bijective mapping from $\bar{x}$ to V_Q, *i.e.,* it assigns a distinct variable to each vertex v in V_Q. For $x \in \bar{x}$, we use $\mu(x)$ and x interchangeably when it is clear in the context.

A *match* of pattern $Q[\bar{x}]$ in graph G is a homomorphism h from Q to G such that (a) for each $u \in V_Q$, $L_Q(u) = L(h(u))$; and (b) for each $e = (u, l, u')$ in Q, $e' = (h(u), l, h(u'))$ is an edge in G.

A *predicate* of pattern $Q[\bar{x}]$ is one of the following:

$$p ::= l(x, y) \mid x.A \otimes y.B \mid x.A \otimes c \mid \mathsf{2WL}_L(x, y, l) \mid \mathcal{M}(x.\bar{A}, y.\bar{B}),$$

where $\otimes$ is one of $=, \neq, <, \leq, >, \geq$; x and y are variables in $\bar{x}$; c is a constant; A and B are attributes; and $x.\bar{A}$ is a list of attributes at "vertex" x; similarly for $y.\bar{B}$. We support (a) *link predicate* $l(x, y)$, indicating the existence of an edge labeled l from vertex x to y; (b) *attribute predicates* $x.A \otimes y.B$ and $x.A \otimes c$ to compare attribute values; (c) $\mathsf{2WL}_L(x, y, l)$, to check whether there is a link of "type" (label) l from x to y by local 2-WL (Weisfieler-Leman) test [15] (see below); and (d) *ML predicate* $\mathcal{M}(x.\bar{A}, y.\bar{B})$ returns true iff $\mathcal{M}$ predicts true at $(x.\bar{A}, y.\bar{B})$; here $\mathcal{M}$ is an ML model that returns Boolean (*e.g.,* $\mathcal{M} \geq \sigma$ if the strength of the $\mathcal{M}$ prediction is above a predefined bound σ).

A *graph association rule* (GAR) φ is defined as

$$Q[\bar{x}](X \rightarrow p_0),$$

where $Q[\bar{x}]$ is a graph pattern, X is a (possibly empty) conjunction of predicates of $Q[\bar{x}]$, and p_0 is a predicate of $Q[\bar{x}]$. We refer to $Q[\bar{x}]$ and $X \rightarrow p_0$ as the *pattern* and *dependency* of φ, respectively, and to X and p_0 as the *precondition* and *consequence*, respectively.

Intuitively, the pattern Q extracts related entities in a graph, and the dependency $X \rightarrow p_0$ is applied to the entities. Attribute

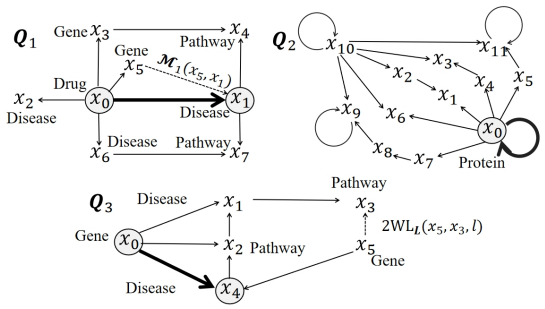

Figure 1: Graph patterns in GARs

predicates $x.A = c$ and $x.A = y.B$ specify *value associations of* attributes, and link predicates $l(x, y)$ make *link associations*. One can "plug in" pre-trained ML models $\mathcal{M}$ for *e.g.,* ER and classification.

Predicate $\mathsf{2WL}_L(x, y, l)$ is used to explain the predictions of $\mathcal{M}$. As shown in [15], most GNN models for link prediction are based on the 1-WL test, and are at most as expressive as local 2-WL test. Thus with $\mathsf{2WL}_L(x, y, l)$, GARs can explain such GNN predictions.

Below we exemplify GARs discovered from real-life data.

(1) Drug repurposing. Upon the request of our biomedical partners, MedHunter was used to discover GARs for repositioning of existing drugs on a type of Parkinson disease. A GAR is $\varphi_1 = Q_1[\bar{x}](X_1 \rightarrow l(x_0, x_1))$ in Figure 1. The rule is discovered from CTD [3]. Together with Q_1, precondition X_1 specifies the following: (1) drug x_0 has a known effect on an inborn genetic blood disease x_2; (2) disease x_1 is the Parkinson; (3) drug x_0 interacts with a gene x_3, which shares an effect pathway x_4 with x_1; (4) drug x_0 can interact with a gene x_5, which has an $\mathcal{M}_1$-predicted relationship with x_1 (the dashed arrow in Q_1), where $\mathcal{M}_1$ is a model that predicts the associations between genes and diseases [18, 20, 21]; and (5) drug x_0 has a known effect on a type of skin cancer x_6, which shares an effect pathway with x_1. The predicted link $l(x_0, x_1)$ (the bold line in Q_1) indicates that drug x_0 may be associated to Parkinson's disease x_1 in some way.

Such GARs found five drugs for Parkinson, four with published evidence and the remaining one is under lab investigation.

(2) Protein-protein interaction. BioGRID [2] is a popular biomedical repository. After enriching it with external data, *e.g.,* UniProt [5], a GAR is $\varphi_2 = Q_2[\bar{x}](\mathcal{M}_2(x_0, x_0) \geq \delta \wedge X_2 \rightarrow l(x_0, x_0))$, in which $\mathcal{M}_2$ is a RGCN model [19] for PPIs, Q_2 is given in Figure 1 in which each vertex (resp. edge) denotes a protein (resp. PPI); here X_2 specifies logic conditions *e.g.,* the domains and subcellular locations. This GAR identifies the self-interaction $l(x_0, x_0)$ (the bold loop in Q_2) on x_0 by filtering the FPs of the prediction of $\mathcal{M}_2$ with additional logic conditions in X_2, *i.e.,* although the strength of the $\mathcal{M}_2$ prediction is above the threshold δ, the prediction is made only if X_2 holds.

A PPI identified by such GARs coincides a finding in [17].

(3) Explanation. If p_0 is an ML predicate, MedHunter can discover GARs to explain the ML prediction, *e.g.,* $\varphi_3 = Q_3[\bar{x}](X_3 \rightarrow \mathcal{M}_3(x_0, x_4))$ in Figure 1, where $\mathcal{M}_3$ is an ML model that predicts the interaction between gene x_0 and disease x_4. This GAR explains why $\mathcal{M}_3(x_0, x_4)$ is true (the bold line in Q_3) with both Q_3 and X_3, *i.e.,* (1) gene x_0 has interacted with pathway x_2; (2) gene x_0 has interacted with disease x_1 that has also interacted with pathway x_3; (3) gene x_5 is predicted to interact with pathway x_3 by $\mathsf{2WL}_L$, and it has effect on disease x_4; and (4) pathway x_2 has effect on diseases x_1 and x_4.

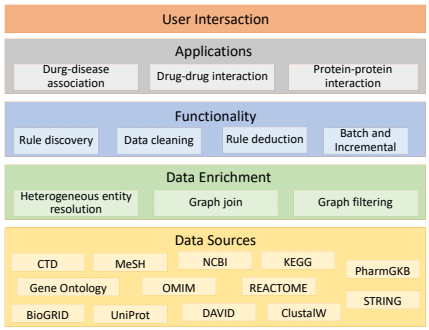

Figure 2: The architecture of MedHunter

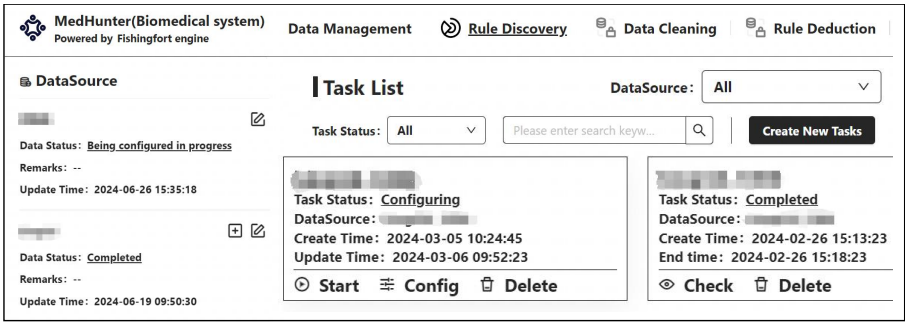

Figure 3: A snapshot of the user panel of MedHunter

Algorithms. MedHunter implements (a) the algorithm of [8] for discovering GARs, (b) an extension of [12] for deducing missing DDIs, DDAs and PPIs; it chases a biomedical graph with a set Σ of mined GARs by accumulating and referencing ground truth in the process. Both algorithms are parallely scalable, *i.e.,* they provably guarantee to reduce runtime when given more processors [16].

2.2 Cleaning and Enriching DDKG

MedHunter maintains DDKG by enriching and cleaning its data.

Drug-Disease Knowledge Graph. DDKG integrates biomedical libraries and data banks, by computing their join as follows.

MedHunter aligns entities across different graphs by employing heterogeneous entity resolution (HER) [9]. For a vertex v_1 in graph G_1 and vertex v_2 in another graph G_2, it determines whether v_1 and v_2 refer to the same entity via parametric simulation, which embeds ML models for similarity checking in topological matching.

Given $G_1 = (V_1, E_1, L_1, F_1)$ and $G_2 = (V_2, E_2, L_2, F_2)$, MedHunter "joins" them as $G_\oplus(G_1, G_2) = (V_\oplus, E_\oplus, L_\oplus, F_\oplus)$, where $V_\oplus$ (resp. $E_\oplus$) is a revision of the union $V_1 \cup V_2$ (resp. $E \cup E_2$) such that u and v are represented as the same (merged) vertex in $V_\oplus$ if (u, v) is a match by HER; and $L_\oplus$ and $F_\oplus$ inherit the label and attribute assignments from G_1 and G_2. When u and v both carry attribute A, the merged vertex takes the value $v.A$ from more reliable G_i ($\in [1, 2]$).

Data enrichment. To ensure that DDKG remains up-to-date and relevant, MedHunter continuously and incrementally enriches it in response to updates ΔG to the source graphs. Rather than adding all data from ΔG, it employs the graph filtering method of [11] to extract only relevant data to DDKG; it applies LSTM to pick "important" paths ranks the paths, extracts data from top-ranked paths, and enriches DDKG with the data. As shown in [11], graph filtering is effective in reducing noise and the size of DDKG.

Data cleaning. MedHunter maintains the quality of DDKG via a (human-in-the-loop) cleaning procedure, using a special case of GARs, referred to as Graph Cleaning Rules (GCRs) [7]. A GCR has a form $Q[x_0, y_0](X \to p_0)$, where $Q[x_0, y_0] = \langle Q_x[x_0, \bar{x}], Q_y[y_0, \bar{y}]\rangle$, $Q_x[x_0, \bar{x}]$ is a star shape pattern with a designed x_0; similarly for $Q_y[y_0, \bar{y}]$. GCRs support all the predicates of GARs except $2\mathsf{WL}_L$.

Intuitively, Q specifies two entities x_0 and y_0 with heterogeneous structures in a schemaless graph. The star patterns identify features of x_0 and y_0. We restrict Q_x and Q_y to a star shape so that it takes polynomial time to check matches of such Q, and to apply GCRs [7].

MedHunter automatically discovers GCRs from DDKG. To clean data with certain fixes, users can selectively apply a set of GCRs based on labeled ground-truth data; they may optionally intervene in a prompt-and-confirm style to facilitate the cleaning process.

Algorithms. MedHunter implements (a) the algorithms of [11] for graph join, data enrichment, and incremental discovery of GARs; (b) the algorithms of [7] for mining GCRs and detecting errors; and (c) an extension of the algorithm of [12] for fixing the detected errors. All the algorithms are also parallelly scalable [16].

2.3 The Architecture of MedHunter

The architecture of MedHunter is shown in Figure 2. It consists of the following main modules, from the bottom to the top.

- *Data sources*, which maintain raw data from various libraries and data banks. New data can be dynamically merged with DDKG.
- *Data enrichment*. On top of the data source module, MedHunter conducts data enrichment, by first aligning entities across different sources and then leveraging graph join and graph filtering techniques to construct a unified knowledge graph DDKG.
- *Rule discovery and data cleaning*. Based on the integrated DDKG, MedHunter supports GAR/GCR discovery, cleaning and association deduction, in both batch and incremental modes. before.
- *Applications*. The GARs discovered are then used in various biomedical applications, for identifying DDAs, DDIs and PPIs,

3 DEMONSTRATION SETUP AND PLAN

This section proposes our plan for demonstrating MedHunter[1].

Setup. We will use a single machine powered by 16GB RAM and 8 processors with Intel(R) Xeon(R) Gold 5320 CPU @2.20GH.

User interface. Participants are invited to access MedHunter from its Web UI. As shown in the snapshot of Figure 3, MedHunter presents a friendly interface. The top navigation bar provides access to three main modules of MedHunter. Each module is task-based, and it takes only a few clicks to get a task up and running. We will walk users through the process of creating new tasks, where they configure inputs and monitor its progress. The task results are visualized and updated in real time through the dashboard.

Data enrichment. We will give a guided tour of how MedHunter enriches DDKG from external data, *e.g.,* PharmGKB [4]. User can "join" data and DDKG, and take the enriched DDKG for further analysis. The process is interactive and visualized on the Web interface.

[1]https://youtu.be/rhvcXBjtLhw

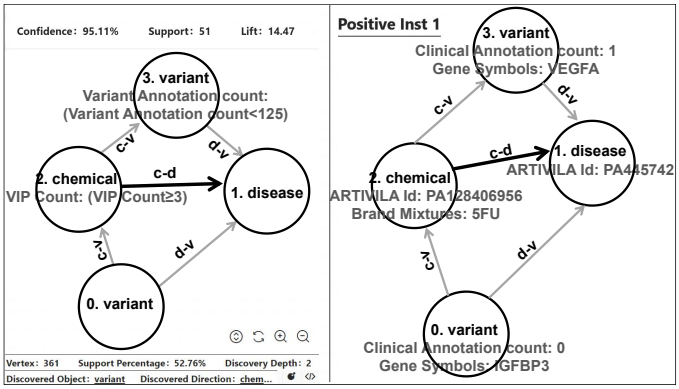

Figure 4: MedHunter in DDAs

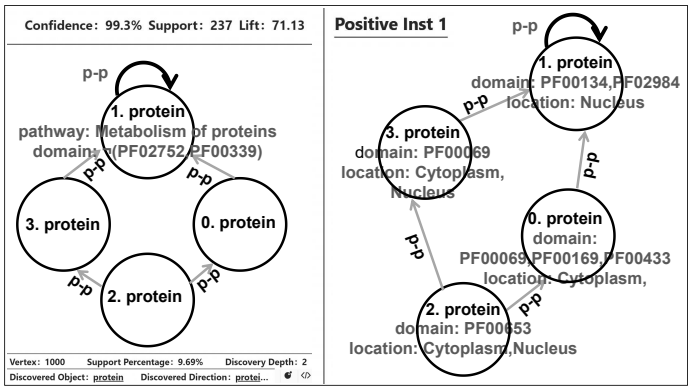

Figure 5: MedHunter in PPIs

Data cleaning. We will walk users through the data cleaning process of MedHunter. We will guide users to configure and run rule discovery for GCRs. The users can examine the discovered GCRs, and pick a subset for cleaning. We will demonstrate interactive data cleaning, using the selected GCRs. During the process, MedHunter will visualize detected errors and the GCRs, with suggested fixes. A user may opt to either accept a suggestion or customize a fix, which will be taken as ground truth for subsequent GCR applications.

Applications. Participants are also invited to experience drug discovery with MedHunter, in the following three scenarios. Each scenario is offered as a task template under rule discovery.

Scenario 1: DDAs. Users may pick a disease y, and identify either (a) molecule targets or (b) existing drugs that may have therapeutic effects on the disease. It is to discover and apply GARs whose consequence has the form $l(x, y)$, where x is a drug or a gene. Over DDKG, we will direct users to specify y in the task template for GAR discovery and monitor its execution. The discovered GARs and their predicted DDAs are visualized and updated in real time.

A learned GAR $\varphi_4 = Q_4[\bar{x}](x_2.\mathsf{N}_{\mathsf{VIP}} \geq 3 \wedge x_3.\mathsf{N}_{\mathsf{anno}} < 125 \rightarrow l(x_2, x_1))$ is shown in the left panel of Figure 4. It suggests that a drug x_2 may be associated to disease x_1, provided that (1) x_2 has a high VIP count (*i.e.,* an active compound), and (2) x_1 and x_2 both interact with a gene variant x_3 that has a low annotation count (< 125). MedHunter reports that GAR φ_4 has a support of 51, with confidence over 0.95. One of its predicted DDAs is between Drug #PA128406956 and Disease #PA445742 (shown in the right panel).

Scenario 2: DDIs. MedHunter can also disclose drug-drug interactions. We will demonstrate this functionality in a procedure similar to DDAs, except that it allows users to specify a target drug as y.

Scenario 3: PPIs. The participants will also witness how MedHunter identifies protein-protein interactions. It is to discover and apply link predicting GARs over the PPI network of DDKG, along with ML models for PPI predictions, *e.g.,* RGCN. We will guide the users to examine the result as GARs focus on reducing the FPs and FNs.

Figure 5 shows a discovered GAR $\varphi_5 = Q_5[\bar{x}](x_1.\text{pathway} = \text{Metabolism\_of\_proteins} \wedge x_1.\text{domain} = \text{Cyclin} \wedge \mathcal{M}_2(x_1, x_1) = \text{false} \rightarrow l(x_1, x_1))$. The rule's confidence is 0.993; it suggests the self-PPI on protein x_1 despite the negative prediction of $\mathcal{M}_2$, if (1) Cyclin-domain x_1 is involved in the pathway "Metabolism of proteins", and (2) x_1 shares at least two common PPIs with protein x_2.

We will show that MedHunter reduces the FPs and FNs of ML models for these tasks, by improving the precision by 4% on average.

ACKNOWLEDGMENTS

This work is supported by China NSFC 62202313 and Guangdong Basic and Applied Basic Research Foundation 2022A1515010120.

REFERENCES

[1] 2024. AI-based Drug Development Solution. https://www.grandhoo.com/en/fishtown/product-solution/qubing/.
[2] 2024. BioGRID. https://thebiogrid.org/.
[3] 2024. Comparative Toxicogenomics Database (CTD). https://ctdbase.org/.
[4] 2024. PharmGKB dataset. https://www.pharmgkb.org/.
[5] 2024. UniProt. https://www.uniprot.org/.
[6] Jin-yi Cai, Martin Fürer, and Neil Immerman. 1992. An optimal lower bound on the number of variables for graph identification. *Comb.* 12, 4 (1992), 389–410.
[7] Wenfei Fan, Wenzhi Fu, Ruochun Jin, Muyang Liu, Ping Lu, and Chao Tian. 2023. Making It Tractable to Catch Duplicates and Conflicts in Graphs. *Proc. ACM Manag. Data* 1, 1 (2023), 86:1–86:28.
[8] Wenfei Fan, Wenzhi Fu, Ruochun Jin, Ping Lu, and Chao Tian. 2022. Discovering Association Rules from Big Graphs. *PVLDB* 15, 7 (2022), 1479–1492.
[9] Wenfei Fan, Ling Ge, Ruochun Jin, Ping Lu, and Wenyuan Yu. 2022. Linking Entities across Relations and Graphs. In *ICDE*. IEEE, 634–647.
[10] Wenfei Fan, Ruochun Jin, Muyang Liu, Ping Lu, Chao Tian, and Jingren Zhou. 2020. Capturing Associations in Graphs. *PVLDB* 13, 11 (2020), 1863–1876.
[11] Wenfei Fan, Muyang Liu, Shuhao Liu, and Chao Tian. 2024. Catching More Associations by Referencing Knowledge Graphs. *PVLDB* (2024).
[12] Wenfei Fan, Ping Lu, Chao Tian, and Jingren Zhou. 2019. Deducing Certain Fixes to Graphs. *PVLDB* 12, 7 (2019), 752–765.
[13] Chris Fotis, Asier Antoranz, Dimitris Hatziavramidis, Theodore Sakellaropoulos, and Leonidas G. Alexopoulos. 2017. Pathway-based technologies for early drug discovery. *Drug Discovery Today* (2017).
[14] Martin Grohe. 2020. word2vec, node2vec, graph2vec, X2vec: Towards a Theory of Vector Embeddings of Structured Data. In *PODS*. ACM, 1–16.
[15] Yang Hu, Xiyuan Wang, Zhouchen Lin, Pan Li, and Muhan Zhang. 2022. Two-Dimensional Weisfeiler-Lehman Graph Neural Networks for Link Prediction. *CoRR* abs/2206.09567 (2022).
[16] Clyde P. Kruskal, Larry Rudolph, and Marc Snir. 1990. A complexity theory of efficient parallel algorithms. *Theor. Comput. Sci.* 71, 1 (1990), 95–132.
[17] Ying Lai, Giorgio Fois, Jose R Flores, et al. 2022. Inhibition of calcium-triggered secretion by hydrocarbon-stapled peptides. *Nature* 603, 7903 (2022), 949–956.
[18] Yu Li, Hiroyuki Kuwahara, Peng Yang, Le Song, and Xin Gao. 2019. PGCN: Disease gene prioritization by disease and gene embedding through graph convolutional neural networks. *biorxiv* (2019), 532226.
[19] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web (ESWC)*. Springer, 593–607.
[20] Juan Shu, Yu Li, Sheng Wang, Bowei Xi, and Jianzhu Ma. 2021. Disease gene prediction with privileged information and heteroscedastic dropout. *Bioinformatics* 37, Supplement_1 (2021), i410–i417.
[21] Xiaochan Wang, Yuchong Gong, Jing Yi, and Wen Zhang. 2019. Predicting gene-disease associations from the heterogeneous network using graph embedding. In *IEEE International conference on bioinformatics and biomedicine (BIBM)*. 504–511.
[22] Xiangxiang Zeng, Xinqi Tu, Yuansheng Liu, Xiangzheng Fu, and Yansen Su. 2022. Toward better drug discovery with knowledge graph. *Current opinion in structural biology* 72 (2022), 114–126.