Querying of Disparate Association and Interaction Data in Biomedical Applications

Shi Qiao, Mehmet Koyutürk, and Meral Z. Özsoyoğlu

Abstract—In biomedical applications, network models are commonly used to represent interactions and higher-level associations among biological entities. Integrated analyses of these interaction and association data has proven useful in extracting knowledge, and generating novel hypotheses for biomedical research. However, since most datasets provide their own schema and query interface, opportunities for exploratory and integrative querying of disparate data are currently limited. In this study, we utilize RDF-based representations of biomedical interaction and association data to develop a querying framework that enables flexible specification and efficient processing of graph template matching queries. The proposed framework enables integrative querying of biomedical databases to discover complex patterns of associations among a diverse range of biological entities, including biomolecules, biological processes, organisms, and phenotypes. Our experimental results on the UniProt dataset show that the proposed framework can be used to efficiently process complex queries, and identify biologically relevant patterns of associations that cannot be readily obtained by querying each dataset independently.

Index Terms—UniProt, RDF, graph template matching, query optimization

1 Introduction

Network models are commonly encountered in systems biology. Interactions among biomolecules are naturally represented as networks, in which an edge between two nodes models a physical, regulatory, functional, or statistical interaction between two or more molecules. Such molecular interaction networks include protein-protein interaction (PPI) networks [1], gene co-expression networks [2], transcriptional regulatory networks [3], metabolic pathways [4], genetic interaction networks [5], and signaling pathways [6]. Network models are also used to represent high-level associations among various biomolecules, as well as other biological entities such as ligands, cellular functions, functional modules, biological processes, tissues, organisms, and phenotypes. Higher level associations represented by networks include gene-disease associations [7], clinical similarity or co-morbidity of diseases [7], disease-drug associations [8], molecular response to drugs [9], functional annotation of genes and proteins, and evolutionary relationships among molecules and organisms [10].

Many algorithms have been developed for integrated analyses of disparate interaction and association data. These algorithms represent multiple types of associations using heterogeneous network models and use these integrated networks to extract previously uncharacterized relationships among biomolecules, their function, biological processes, diseases, and drugs. The applications of such integrative models include functional characterization of genomic variants associated with complex diseases [11], disease gene prioritization [12], drug repositioning [13], characterization of pathogen-host relationships [10], and functional annotation of proteins [14]. However, most of the databases that provide interaction and association data focus on a specific data type and tools for integrated querying of different types of interactions and associations are highly limited. For these reasons, studies on integrative analysis of network data are conducted by downloading different datasets in bulk and incorporating dedicated algorithms to integrate these datasets for specific analysis and mining tasks. In other words, it is not straightforward for a researcher to identify or infer indirect associations among biological entities by incorporating data in multiple forms.

In this study, with a view to facilitate exploratory querying of integrated biological networks, we develop an RDF (Resource Description Framework)-based querying framework. Here, the term "integrated biological network" refers to the collection of physical, regulatory, functional, and statistical interactions, as well as associations among biological entities. RDF is the first W3C standard for enriching information resources on the web with detailed descriptions (i.e., Metadata) [15]. It is the commonly used data model for linked data, and knowledge bases that are shared and exchanged on the web. An RDF dataset consists of a set of triples, in the form (s, p, o), where s represents a subject, p represents a property, and o represents an object. The triplet (s, p, o) states that subject s has the property p whose value is the object o. In RDF models, unique identifiers (URI's) are used for subjects, properties or objects to uniquely refer to entities, relationships or concepts. Literals can also be used for objects [15]. Due to its simplicity and generalizability, RDF enables seamless interoperability and integration of the data on the web. For this reason, it is commonly used to represent a wide range of data and information

- *S. Qiao is with Microsoft Corporation, One Microsoft Way, Building 17, Redmond, WA 98052. E-mail: shqiao@microsoft.com.*
- *M. Koyutürk and M.Z. Özsoyoğlu are with the Department of Electrical Engineering & Computer Science, Case Western Reserve University, 10900 Euclid Ave., Olin 512, Cleveland, OH 44106. E-mail: {mxk331, mxo2}@case.edu.*

Manuscript received 31 Mar. 2016; revised 18 Oct. 2016; accepted 29 Nov. 2016. Date of publication 8 Dec. 2016; date of current version 6 Aug. 2018.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCBB.2016.2637344

from structured, semi-structured, or unstructured sources. RDF data can be abstracted and visualized using graph models. In such a model, nodes represent subjects and objects, and edges represent properties (predicates). Since each edge in this model corresponds to a binary predicate, this model can be used for reasoning and inference based applications.

Existing resources for biological interaction and association data (e.g., UniProt [16], IntAct [17], Reactome [18], OMIM [19]) enable querying of these data by using a relational database. However, creating a knowledgebase using a relational database requires parsers for each dataset, designing a schema, and tuning the system. Due to rapid acquisiation of knowledge, the file formats and schema in these knowledge bases need to be updated frequently. These changes require costly updates for the parsers, the schema, and the queries to keep the system up-to-date and functional. In comparison, using an integrated RDF dataset offers several advantages. First, using RDF, queries do not depend on the schema as in relational databases. Second, RDF enables seamless data integration in which all interactions and associations are represented as triplets. The generalizability of RDF can be exploited to integrate data coming from various sources and the resulting datasets can be queried directly for interactions and associations between disparate biological entities.

Here, we propose a graph template-based query framework for querying biological associations and "interactions" from an RDF knowledge base (Based on the datasets we used and for simplicity, we don't distinguish protein/gene interaction from co-expression relations, functional relations, or statistical relations). In this framework, queries are expressed as a graph template, in which the nodes represent biological entities involved in the query, and edges represent interactions and/or associations between these entities. The main contribution of this study is the development of a tool that provides a "shortcut" to data integration in the bioinformatics pipeline. By providing a powerful mechanism for seamlessly querying an RDF dataset integrating various types of biological interactions and associations, this framework enables simple querying and efficient processing for highly sophisticated queries that can provide novel biological insights. We comprehensively test the proposed framework on data obtained from UniProtKB, using a large set of grap template-based queries representing a variety of biological research questions. Our results show that the proposed framework can process highly sophisticated queries efficiently, with real-time performance in most cases, and the results of these queries can be used to generate novel biological hypotheses.

The rest of the paper is organized as follows. In Section 2, we provide examples of using graph template queries for extracting associations and interactions among biological entities that are studied by biomedical scientists. Section 3 provides an overview of RDF data model and Querying RDF using graph templates. In Section 4, we describe the UniProt knowledge base and its components that are utilized in this paper. In Section 5, we present the experimental analysis, queries, and query results. We discuss the related work in Section 6, and conclude in Section 7.

2 Query Examples and Motivation

In biomedical research, researchers could use public databases for various purposes, including interpretation of findings, and exploratory data analysis for the identification of novel relationships. Existing databases provide query interfaces for "first order" queries, such as listing interacting partners of a query protein, identifying protein families that contain amino acid sequences homologous to a query sequence, listing genes that are associated with a query disease, and so on. Clearly, systems-level analyses require more sophisticated and advanced queries. A commonly encountered query in this regard involves functional annotation of a set of genes/proteins that are identified to be of interest in an experiment (e.g., a set of differentially expressed genes between two different phenotypes [20] or a set of differentially expressed proteins in a specific stage of cancer [21]). While there are many algorithms developed for this purpose (namely, enrichment analysis), many other types of queries are also of interest for domain scientists.

In most cases, existing databases do not provide query interfaces for making indirect inferences across different types of associations. However, the current state-of-the-art in integrated mining of biological data shows that such indirect inferences can help extract novel information that cannot be obtained by separated querying of each database [12], [13], [14]. Building on the success of these integrative data mining efforts, we propose queries that integrate multiple types of association and integration data which enable scientists to effectively explore indirect relationships among biological entities. Such queries include the following:

- *Identification of orthologous phenotypes:* "Phenologs" are defined as phenotypes related by the orthology of the associated genes in two organisms [22]. Phenologs are the phenotype-level equivalent of gene orthologs. Such orthologous phenotypes can be identified through integrated analysis of gene-phenotype associations and sequence homology between genes that belong to different species. Phenologs enlight insights into unforeseen relationships among different phenotypes [23]. Therefore, enabling the biomedical researchers to search for "phenologs" of a set of genes can be useful for exploratory analysis that exploits evolutionary relationships. Fig. 1f shows one query template that addresses this question. The user specifies two organisms and a query phenotype (QP) in one of the organisms and tries to identify phenotypes (TP) in the second organism associated with proteins that belong to the same protein family as the proteins associated with the query phenotype in the first organism.
- *Overlap in the molecular bases of diseases:* Shared molecular bases of diseases provide hints on potentially uncharacterized relationship between different diseases. The overlap in the identity of genes that are associated with each disease is commonly utilized to identify shared molecular bases [7]. Inspired by this observation, integration of disease association data across different diseases provide useful exploratory queries. Namely, the user can specify a disease and search for all diseases that share at least a (statistically significant) number of gene associations with a disease of interest. Such a question can be addressed by a graph template matching query shown in Fig. 1a. In this query, the user specifies a disease (QD) and queries for all diseases (TD) such that a protein associated with QD is also associated with TD.

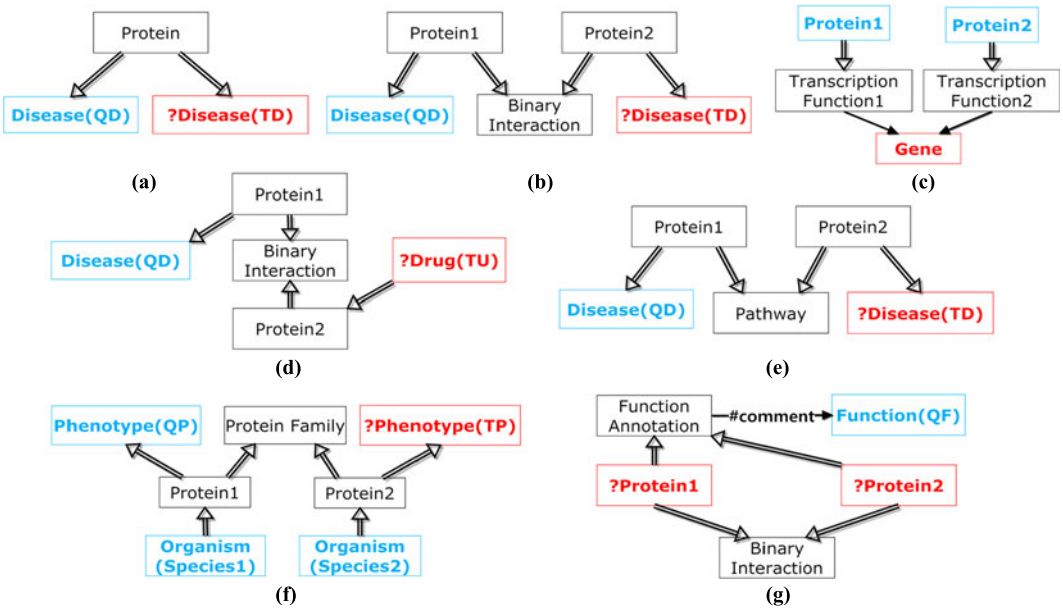


Fig. 1. Query examples and motivation (blue nodes represent user specified entities, e.g., "QD" and "QF"; red nodes with "?" represent query output entities, e.g., "?TD" and "?TU"; double line edges represent path connections).

- *Shared molecular interactions between diseases:* The overlap between the molecular etiologies of related diseases is necessarily limited to the identity of gene associations. These relationships may also be uncovered through molecular interactions among the genes associated with different diseases [24]. Therefore, identification of shared interactions between genes that are implicated in different diseases provides systems level insights into the shared molecular mechanisms of these diseases [25]. Such relationships can be discovered using a query like the one shown in Fig. 1b. In this query, the user specifies a disease (QD) and queries for all diseases (TD) such that a protein associated with QD interacts with a protein associated with TD.
- *Shared molecular interactions between diseases and drugs:* Since repurposing drugs dramatically reduces the costs in drug development, pharmaceutical companies are very interested in exploiting the growing wealth of omic data for drug repositioning [26]. In this application, multiple types of omic data are integrated with data on drug response to investigate functional relationships between biomolecules targeted by drugs. These relationships are then used to match disease and drugs based on shared molecular connections. Therefore, queries seeking to identify drugs that share a number of molecular interactions with a given disease (i.e., molecular interactions between the targets of the drug and the genes associated with the disease) can produce an excellent starting point for identifying candidate drugs for repositioning [8]. A graph template matching query that can be used for this purpose is shown in Fig. 1d. In this query, the user specifies a disease (QD) and queries for all drugs (TU) such that a protein associated with QD interacts with a protein associated with TU.
- *Network schemas:* Integrated mining of molecular interaction networks and functional annotations led to the identification of network schemas, i.e., small subgraphs of functional terms that recur frequently in molecular interaction networks [27]. These network schemas provide insights into conserved functional modules and the design principles of cellular networks [28]. A graph template matching query such as the one in Fig. 1g can be used to search for network schemas involving specific biological processes or molecular functions. In this query, the user specifies a specific molecular function (QF) and queries for all proteins associated with QF interacting with each other.
- *Shared pathways between diseases:* Organization of systems biology knowledge in the form of pathways provides well-established, reliable, and tractable access to state-of-the-art knowledge on biological processes. For this reason, incorporation of pathway data in analyzing the relationship among phenotypes provides information that is complementary to high-throughput interaction data that is organized into networks [29]. For example, identification of shared pathways among different diseases is useful in understanding similarities in disease development and progression [28]. A sample graph template query that can be used for this purpose is shown in Fig. 1e. In this query, the user specifies a disease (QD) and queries for all diseases (TD) such that a pathway contains a protein associated with QD also contains a protein associated with TD.
- *Common down-stream effects of perturbations on signaling.* Experimental approaches to characterizing the functional consequences of molecular perturbations are usually based on interfering with the target molecule (e.g., via gene knock-outs or RNA interference) and then measuring the changes in a specific phenotypic trait or the abundance/activity of other molecules [30], [31]. In the interpretation of the results of such experiments, an important question is the characterization of common down-stream effects of perturbations with correlated phenotypic outcome. For example, for signaling proteins, strong positive or negative correlation

between two proteins' influence on the phenotype may be indicative of common downstream effects [32]. With a graph template query on an integrated database of protein-protein interactions and transcriptional regulatory interactions, scientists can quickly discover potential candidates for these common effects by querying for paths that go through these proteins and converge into the same node in the integrated network. A sample query that can be used for this purpose is shown in Fig. 1c. The user specifies two proteins of interest and aims to identify the common gene regulated by both proteins through transcriptional regulatory interactions.

As demonstrated by these examples, the ability to seamlessly query interactions and associations among biological entities enables biomedical scientists to run exploratory queries spanning a broad range of knowledge bases. These queries enable scientists to explore a broad range of sophisticated questions on the relationships between these entities. Today, the most common way of interpreting observed experimental associations among a group of proteins is to identify subnetworks that connect the proteins of interest, e.g., using Steiner tree based algorithms [33]. Biomedical researchers often use commercial software (e.g., Ingenuity Pathway Analysis, Pathway Studio) that do not provide algorithmic transparency or clearly defined criteria for the identified subnetworks. Cytoscape provides a free and accessible tool for visualizing and analyzing various biological networks; however, it is not built on a database system and requires the user to download the data sets and integrate these data through plug-ins for in-memory analysis tasks [34]. To this end, semantically meaningful queries that i) are connected to a database server, ii) integrate multiple data types, and iii) can be specified using graph templates, and, can generate significant mechanistic insights into the relationships among proteins of interest and provide the researchers with new ways of thinking about their research questions.

While all of the sample queries listed above can be currently processed by downloading bulk datasets from multiple databases and subsequent processing and joining of these datasets, this is often time-consuming and challenging for many biomedical scientists. Therefore, the main contribution of this study is the development of a querying framework that provides a "shortcut" to data integration in the bioinformatics pipeline. This is useful for two types of applications: 1) "targeted" queries, that is when the researcher is interested in identifying new associations for one or more specific entities (e.g., a group of genes, a particular disease, a particular drug). 2) "high-throughput" queries for mining tasks that is when the researcher is interested in identifying all associations that exhibit a specific pattern (e.g., all disease-drug pairs that share a reasonably large number of interactions).

3 RDF Query Framework

In this section, we first provide basic definitions. We then describe the indexes we use to facilitate graph-template based querying on an RDF database. Finally, we describe the proposed RDF query framework.

A preliminary version of the RDF query framework is described in [35]. Here, with a view to improving the proposed framework, we redesign the indexes and propose new algorithms for neighborhood check and connecitivity check processes. The new query framework is designed to achieve two critical improvements compared to the framework described in [35]: 1. faster connectivity check; 2. lower index space requirements. We explain these modifications in detail in this section.

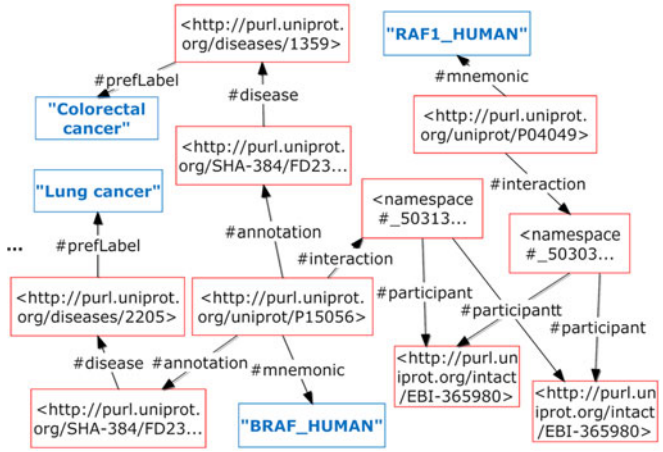


Fig. 2. UniProtKB RDF graph example.

3.1 Definitions

An *RDF Graph* is a directed graph $G = \{V, E, l, f\}$ where V is a set of vertices representing either subjects, objects or both. $E \subseteq V \times V$ is a set of directed edges representing predicates pointing from subjects to objects. l is a label set for subjects, objects and predicates. $f : V/E \rightarrow l$ denotes the mapping function between vertices/edges and labels.

A sample UniProt RDF dataset represented as a graph is shown in Fig. 2. This graph represents the relationships between two proteins (BRAF and RAF1) and two diseases (Lung cancer and Colorectal cancer). In general, proteins are associated with diseases through disease annotation nodes, which are derived from the OMIM database. Protein-Protein interactions are obtained from the IntAct database. Note that, for a binary protein–protein interaction to be included in the Uniprot dataset, two IntAct EBI resources need to be identified. This is due to the quality requirements of UniProtKB that only binary interactions which are experimentally supported by multiple observations are imported from the IntAct dataset. By using this integrated database, we demonstrate RDF representations of the association and interaction among different biological entitles from various datasets.

A *GBE (Graph by Example) Query Template* is a directed graph $G_q = \{V, E\}$. G_q specifies a query describing the subgraphs sought in G, in terms of structural and label contsraints. The vertices (objects or subjects) of G_q are labeled by partial keywords (that are substrings of labels in the label set l of RDF graph G. There are two types of edges in G_q : (i) simple edges which represent predicates and directly map to edges in the database graph G, (ii) connection edges representing paths between nodes. More specifically, a *connection edge* ($\overset{E}{\Leftrightarrow}$) represents a path $\omega_{i,j}$ between two nodes n_i and n_j. Expression E describes the distance constraints of $\omega_{i,j}$ (distance is defined as the length of the shortest path between n_i and n_j in G). For example, in Fig. 8c, the edge between node "<http:// purl.uniprot.org/uniprot*" and node "Lung Cancer" is a connection edge representing a path of length 4 or less.

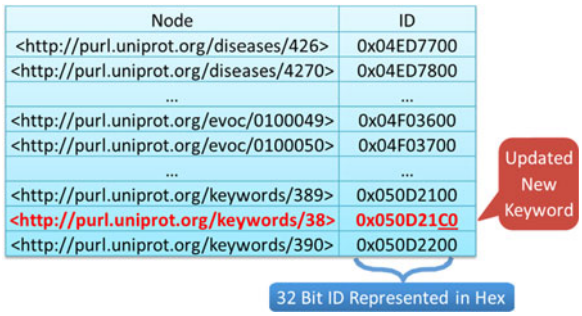


Fig. 3. ByteMap index.

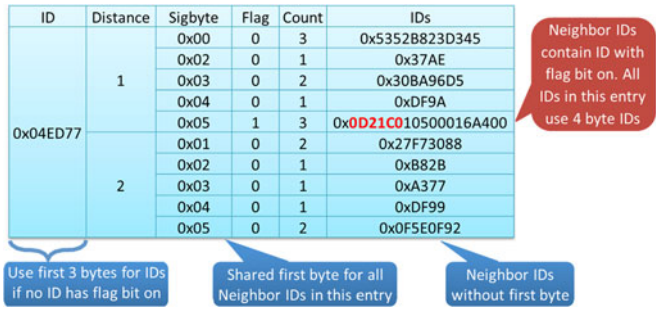


Fig. 4. Neighborhood byte index.

Given RDF graph $G = \{V, E, l, f\}$ and query template $G_q = \{V,\ E\}$, the objective of *Template Matching* is to find all subgraphs of G that satisfy both structural and label constraints in G_q based on graph isomorphism.

An example GBE query template is shown in Fig. 8c. This query finds any cancer associated with a protein that is also associated with "Lung Cancer" (this query is motivated by Fig. 1a and corresponds to Q3 in Section 5). Evaluating this query over the RDF dataset shown in Fig. 2 will return "Colorectal Cancer" as the answer since the protein BRAF is the protein associated with both diseases.

3.2 Indexes

To achieve high in-memory querying performance, we employ the ByteMap and Neighborhood Byte (NB) indexes inspired by the BitWeaving [36] technique.

The ByteMap Index is illustrated in Fig. 3. This index maps RDF labels into byte IDs in lexicographic order. In other words, if label l_i comes before the label l_k in lexicographic order, the ID of l_i should be smaller than the ID of l_k. Each label ID contains 4 bytes (representing a 32-bit integer). The first three bytes in the ByteMap are used to store regular IDs, allowing indexing of RDF graphs with up to 16 million nodes. The last byte is reserved for handling updates. However, if the RDF graph has more than tens of millions of nodes, the last byte can also be utilized to handle extremely large graphs.

The last byte in the Bytpe Map is used to handle updates as follows. Its first bit is utilized as the *flag bit* to indicate the utilization of the last byte. When a new node is inserted into the ByteMap, we locate the node label with the largest ID coming before it in lexicographic order. Then, we set the flag bit in the last byte as 1 and use the last byte to assign an ID. If multiple nodes need to be inserted in the same interval, we sort them in lexicographic order and assign the last byte to them in equal intervals (i.e., new assigned consective IDs are equally distributed). This strategy reduces the likelihood of data reshuffling triggered by conflicting IDs.

The NB (Neighborhood byte) index is similar to the NI index in [35]. For each node in the graph, the NB index materializes the results of a breadth-first search (BFS) up to a specified number of hops. Thus the term neighbor is used in a general sense here, i.e., it the neighbors of a node that are indexed may also include nodes that are more than one hop away from the source node. In the NB index, the number of hops between a node and its neighbors is stored in the Distance field. The index is also organized according to the distance field. The maximum indexed distance is represented by a parameter denoted d_{max}. The NB index is illustrated in Fig. 4.

The NI index groups the neighbors of a node into ID intervals using a *binning factor* m [35]. In contrast, the NB index groups the IDs of neighbors by their most significant byte as Sigbyte. Each index entry for node $n_i \in G$ contains six columns: ID of n_i (if the *flag bit* is 0, the first three bytes of the byte ID is utilized), Distance, number of indexed neighbor nodes, the most significant bytes of the IDs of neighbor nodes, flag bit, and the remaining bytes of neighbor node IDs (without Sigbyte). The field Distance contains the length of the shortest path from n_i to the indexed neighbor node. The neighbor nodes sharing the same distance and significant byte (first byte of the byte ID) are grouped in one index entry, ordered by their IDs. The flag of an index entry is set to 1 if any ID of the neighbors utilize the last byte of the byte ID. Otherwise, the flag of an index entry is set to 0 and all neighbor IDs utilize the first three bytes of the byte ID.

Compared to the NI index [35] which uses 4-byte integers to index each node, the NB index is more space efficient for three reasons: 1. most node IDs indexed in NB index only use the first three bytes of the four-byte ID; 2. the significant byte is shared by all neighbor nodes in the same entry and is therefore indexed once; 3. the IDs of NB index are of fixed length, therefore they do not require any delimiters to distinguish the IDs of the neighbors listed in the IDs column.

3.3 Query Framework

RDF query framework consists of the following steps:

1. Decomposition of query template into connection edges and components without connection edges.
2. Candidate generation and pruning. Namely, we use the ByteMap index to generate matching candidates for each query node and selectively use the NB index for neighborhood check to prune the candidates.
3. Decomposition of each component into a set of smaller basic querying units. For this purpose, we use one level directed trees, named D-trees.
4. Candidate generation for each component. All matching candidates for each D-tree of a component are generated, and joined to find matching results for a component.
5. Connectivity check: Connection edges between components are processed using NB index to generate the final matches for the query template.

Neighborhood Containment Check (step 2), Component Matching (steps 3 and 4), and Connectivity Check (step 5) are discussed in detail below.

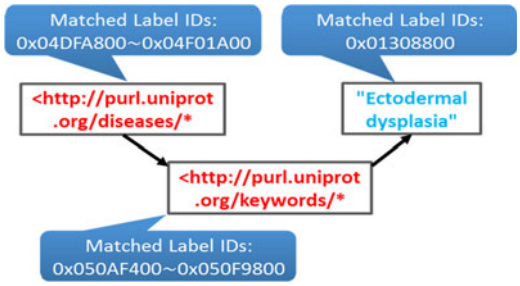


Fig. 5. Example uniprot query.

Algorithm 1. Neighborhood Check (n_i, q_j)

Input: $n_i \in V(G)$, $q_j \in V(G_q)$, ID Intervals $\mathbb{Z}^*$, {Distance, Count} ψ_j^*, NI Index NI_i for node n_i
Output: If n_i pass neighborhood check of q_j, return **true**; Otherwise, return **false**

```
1  FOR all k where |k| ≤ d_max
2    FOR any partial keyword P_k
3      FOREACH value pair {d, c} in ψ_j^k
4        Exact all entries from NI_i where ID interval intersect
         with Z_k and Distance ≤ d
5        Count all IDs in Z_k as c'
6        IF c' ≤ c, RETURN false
7  RETURN true
```

3.3.1 Neighborhood Containment Check

The neighborhood containment check process for NB index utilizes the hierarchy of byte IDs to achieve better performance inspired by the Vertical Bit-Parallel layout of BitWeaving technique (here, we assume partial keywords are specified as prefixes of node labels).

!The *Byte Interval*, $\mathbb{Z}_j$, of a partial keyword $\mathcal{P}_k$ for any query node $q_j \in V(G_q)$, is defined as the set of byte IDs of node set N_j where $\forall n_i \in N_j$, the label l_i of n_i is a valid match for $\mathcal{P}_k$.

!The *K-Neighbor* of a node $n_i \in V(G)$, denoted as $Neighbor_k(n_i)$, is the set of nodes that are forward or backward neighbors of n_i via paths of length k or less. That is, if k is positive, $\forall\, n_j \in Neighbor_k(n_i)$, there is a directed path from n_i to n_j with no more than $|k|$ hops (n_j is a forward neighbor of n_i); if k is negative, $\forall\, n_j \in Neighbor_k(n_i)$, there is a directed path from n_j to n_i with no more than $|k|$ hops (n_i is a forward neighbor of n_j).

!Now we define the Neighborhood Check: Given a node $n_i \in V(G)$, and a query node $q_j \in V(G_q)$, $\mathrm{n_i}$ passes the *Neighborhood Check* of q_j if $\forall\, q_k \in Neighbor_k(q_j)$, ID Interval $\mathbb{Z}_\mathrm{k}$ uniquely contains ID of any $n_g \in Neighbor_k(n_i)$, for all $|k| \le d_{max}$. The implementation of the neighborhood check process utilizing the ByteMap index is shown in Algorithm 1. First a byte interval is formed for each partial keyword in the query template. Since query templates can contain query nodes with the same partial keyword, value pairs as {Distance, Count (total appearance within Distance)} for each partial keyword are maintained for each query node. The neighborhood check is then performed based on partial keywords one by one, and the count of occurrences of this partial keyword is taken into consideration. The term "uniquely contains" in the Neighborhood Check definition means that the node n_g cannot be used to match more than one ID interval. If one partial keyword contains another partial keyword, count in value pairs is updated. In Algorithm 1, {Distance, Count} pairs associated with query node q_j and partial keyword $\mathcal{P}_k$ are denoted as ψ_j^i.

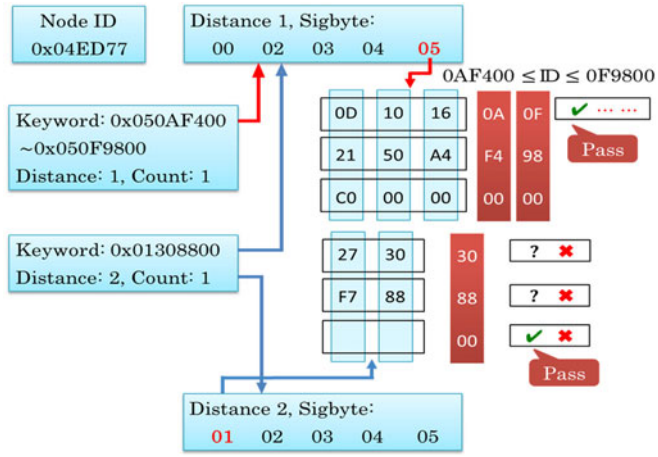


Fig. 6. Neighborhood check.

We also demonstrate the neighborhood check process with an example in Fig. 6. In this example, the neighborhood check is performed for the candidate node 0x04ED77 (<http://purl.uniprot.org/diseases/*>) for the query template in Fig. 5. The candidate node 0x04ED77 should have one neighbor which matches partial keyword <... /keywords/*> within 1 hop, and have one neighbor which matches keyword "Ectodermal dysplasia" within 2 hops in order to pass the neighborhood check process. To consider the partial keywords, we first check the significant byte of all neighbors of node 0x04ED77 and identify the significant byte intersecting with the byte intervals of the partial keywords. As shown in Fig. 6, similar to the BitWeaving technique, the neighborhood check process based on NB index can achieve early termination (e.g., as it identifies one match in the first byte scan of the neighbor IDs for partial keyword <... /keywords/*>).

3.3.2 Component Matching

Matching candidates for each component are found by first decomposing the components into D-trees, then joining the candidates that a matching these D-trees. The runtime of component matching is proportional to $\prod_{\mathrm{i=1}}^{\mathrm{K}} |\mathrm{C_{t_i}}|$ where K is the number of all decomposed one level D-trees and $|\mathrm{C_{t_i}}|$ is the number of matching candidates for each D-tree. Finding D-tree decomposition with minimum number of D-trees is likely to improve the runtime of this process; however, it is equivalent to the vertex cover problem [37]. We use a 2-approximation algorithm, similar to the vertex cover approximation, to generate D-tree decomposition. Basically, an edge $(\mathrm{q_i}, \mathrm{q_j})$ is picked recursively from the query component, and D-trees rooted at $\mathrm{q_i}$ and $\mathrm{q_j}$ are added to the result. We define selectivity value function $\mathrm{S(q_j)} = \mathrm{deg(q_j)}/|\mathrm{C_{q_j}}|$, which takes both query node's degree and its corresponding candidate set size into the consideration as a good measurement of the priority to be selected as root nodes for two reasons: (i) choosing higher degree nodes first is likely to yield better results since D-trees rooted at these nodes can cover more edges in the query component which leads to a smaller K value; (ii) choosing nodes with small candidate

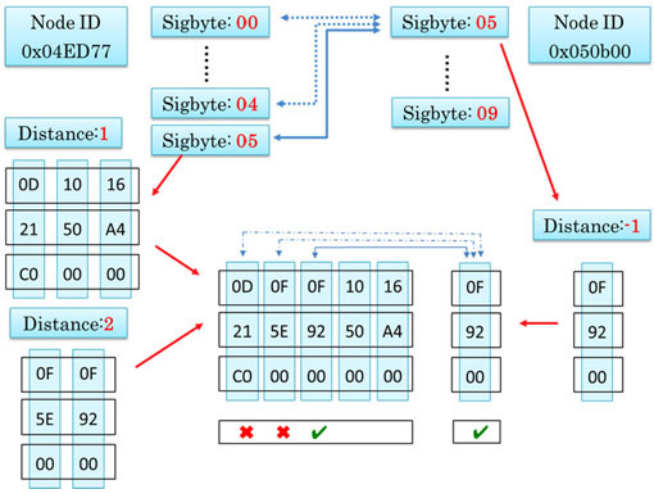


Fig. 7. Merge check.

sets first is likely to yield a lower number of matching candidates for a D-tree. In the second step, NB indexes for all possible root nodes of a decomposed D-tree are checked to generate all D-tree candidate matches. The last step is to join all D-tree candidates together to form component matches. We define the join process as $\mathrm{join}(\mathrm{C_i}, \mathrm{C_j})$, where $\mathrm{C_i}$ and $\mathrm{C_j}$ are candidate sets for two subgraphs of $\mathrm{G_c}$ (it can either be a decomposed D-tree or joined D-trees). $\mathrm{Join}(\mathrm{C_i}, \mathrm{C_j})$ combines each pair of matches from two candidate sets by evaluating the predicate: all shared query nodes of two candidate matches need to have equal matching IDs to join. In order to improve the join performance, a new join order $\mathrm{J_T}$ for the decomposed D-trees is used as follows: 1. begin with D-tree $\mathrm{t_i}$ with smallest candidate set and add $\mathrm{t_i}$ to $\mathrm{J_T}$; 2. add D-tree $\mathrm{t_j}$ with smallest candidate set to $\mathrm{J_T}$ which connects to any already selected D-trees in $\mathrm{J_T}$. Component matching used here is similar to the one used in STWIG [37] with the following differences: 1) D-trees are used as basic join units; 2) a new selectivity function is defined based on the size of candidate sets; 3) the NB index is used to generate all D-tree candidates; 4) tree join order is determined by the sizes of tree candidate sets.

3.3.3 Connectivity Check

Connectivity check verifies the paths in the data graph between the nodes that are connected by connection edges in the query graph. If the connection edge is between nodes of the same component, then the connectivity check is used to prune the candidates of that component. Otherwise, the connectivity check is used to determine whether the two component candidates can join or not. For component connection edges that are within a component, the number of connectivity checks is equal to the size of the component candidate set. For a connection edge between components, the number of connectivity checks depends on the product of the sizes of components' candidate sets. In the worst case, if we have a sequence of N components to be joined by connection edges, the number of connectivity checks that need to be performed can be as large as $\prod_{i=1}^{N} |C_{G_{c^*}}|$. In order to improve query performance, two rules are utilized to determine the order tprocessing connection edges: 1) connection edges inside components are processed before connection edges between components; 2) connection edges between components are processed in the order of those with the smallest product of candidate sets first. NB index is utilized in processing connection edges.

Algorithm 2. MergeCheck (C_{G_c}, i, j, d_c, NB)

Input: Candidate Set of G_c: C_{G_c} , Connection edge between q_i and q_j, Distance Constraint d_c, NB Index NB
Output: Candidate Set $C_{G_c}{}'$ pass Connectivity Check

```
1   Foreach candidate match c in C_Gc
2     If (Merge(c[i], c[j], d_c, NB))
3       Add c to C_Gc'
4   Return C_Gc'
 bool Merge (n_i, n_j, d_c, NB)
5   Index Entries of n_i, n_j as NB_i and NB_j
     (order by Entry.Sig, Entry.Dis)
6   e_i= NB_i.count , e_j= NB_j.count , l = 0, m = 0,
     d_1 = Ceil(d_c/2), d_2 = Ceil(d_c/2) − d_c
7   While(l < e_i & m <e_j)
8     While(l < e_i & NB_i[l].Dis > d_1 & NB_i[l].Dis < 0)
9       l++
10    While(m < e_j & NB_j[m].Dis > 0 & NB_j[m].Dis < d_2)
11      m++
12    If (NB_i[l].Sig < NB_j[m].Sig)
13      l++
14    Else If (NB_i[l].Sig > NB_j[m].Sig)
15      m++
16    Else If (NB_i[l].Sig = = NB_j[m].Sig)
17      Add NB_i[l]. IDs to L_i, Add NB_j[l]. IDs to L_j
18      While(NB_i[l].Sig = = NB_i[l + 1].Sig
         & NB_i[l + 1].Dis ≤ d_1)
19          Add NB_i[l + 1]. IDs to L_i
20          l++
21        While(NB_j[m].Sig = = NB_j[m + 1].Sig
           & NB_j[m].Dis < 0)
22            Add NB_j[m + 1]. IDs to L_j
23            m++
24        Equal Scan L_i with L_j as ψ_{l,m} (using VBP)
25        If (ψ_{l,m}! = ϕ)
26            Return true
27        Else l++, m++
28  Return false
```

The Connectivity check of connection edge between n_i and n_j is performed by retrieving neighbor IDs of n_i and n_j from NB indexes, and then checking whether the neighbors of n_i intersect with the neighbors of n_j. We propose the Merge Check algorithm (Algorithm 2) to provide an efficient index scan process to check whether the neighbors of nodes n_i and n_j are intersecting based on the distance constraint (see Fig. 7 as an example). The index entries of nodes n_i, n_j are ordered by the significant byte and distance in the neighborhood byte index. The neighbor IDs in each index entry are also sorted. By this design, the neighbor ID equality check between two lists of index entries can be performed through a hierarchical merge process (lines 7-20). For each index entry, we first check whether the entry is in the distance constraint (lines 8-11). Then, we check whether the most significant bytes of the two entries are equal. If not, we skip the entry with smaller most significant byte (lines 12-15). If the two entries are equal on the most significant byte, we then check all bytes of neighbor IDs in order of significance. If all

bytes of two neighbors are equal, we return true. If there are no more index entries in either list, we return false to indicate no intersecteing neighbor is found.

The merge check algorithm exploits the benefits of utilizingthe four byte IDs in NB index. By grouping the neighbor IDs according to their most significant bytes, the equality check can skip all neighbors in an entry if the most significant bytes of two index entries are not equal. As the neighbor IDs of the index entries are stored using the Vertical Bit-Parallel layout, equality scans for neighbor IDs are processed similarly as the BitWeaving technique, which achieves early termination.

4 Datasets

The UniProt Knowledgebase (UniProtKB) [16] is a central hub of comprehensive, high-quality, and freely accessible database of protein sequence and functional information. This database provides an integrated view of associations and interactions of proteins with a broad range of other biological entities. One important virtue of UniProtKB is that it allows users to query the related but dispersed information across disparate protein related datasets. Each protein entry recorded in UniProtKB provides a variety of information related to the respective protein, including protein and gene names (mnemonic name, structured name, and alternate names), protein sequences, protein function, catalytic activity, co-factors, subcellular localization, and patterns of expression, protein–protein interactions, and disease association. Besides the rich information provided for each protein, frequent updates and availability in different formats are other advantages of UniProtKB. UniProtKB data is released every 4 weeks to provide the most up to date protein information in multiple formats, including plain text, XML, RDF and GFF.

We utilize UniProtKB in our experiments to take advantage of its high quality and accuracy of data integration. RDF format has no pre-defined schema and RDF is designed to integrate data with ease by combining the triples from different sources directly if unified resource identifiers are utilized. In UniProtKB, each entry undergoes both automated and manual checks to ensure a high level of accuracy and consistency, as well as minimal level of redundancy. The automated check is performed through a quality control software that ensures the correctness of syntax and verification of different biological rules for the entry. Besides this, UniProtKB contains high-quality computationally analyzed records, which are enriched with automatic annotation. The manual review process provides an extra layer of quality control by ensuring that all relevant literature, annotation and analysis results are included. Since we are interested in data integration and the correctness of our queries across multiple datasets is determined by the lowest quality data integrated, the high quality standards provided by UniProtKB are essential to provide high confidence of querying results in the experiments.

4.1 Integrated Datasets

In this section, we describe the three integrated datasets in UniProtKB which are extracted and queried in our experiments: IntAct (protein-protein interaction), Reactome (pathway), and OMIM (disease and phenotype).

4.1.1 IntAct

IntAct [17] provides open-source molecular interaction data populated by interactions curated from the research literature, as well as from direct data depositions. The information within the IntAct database primarily consists of protein–protein interaction (PPI) data. An important virtue of the IntAct dataset is that each entry in IntAct is peer reviewed by a senior curator, and not released until it is accepted by that curator. UniProtKB database is readily integrated with the IntAct database to provide protein–protein interaction data. In order to meet the required quality standard of UniProtKB, only a subset of high quality interactions are imported from IntAct based on a statistical scoring system. A score threshold is chosen by UniProtKB to exclude binary interactions supported by only one experimental observation. In addition to the score-based filter, a set of defined rules are utilized to exclude certain types of data, such as interactions observed in larger complexes, or interactions that have not been experimentally validated. By using these strict criteria, only experimentally validated binary interactions supported by multiple observations are qualified to import into UniProtKB.

4.1.2 Reactome

Reactome [18] is a manually curated open-source biomolecular pathway and reaction dataset. In order to provide a unified identifier, Reactome merges pathway identifier mapping, as well as enrichment and expression analysis tools into a single portal. Reactome uses UniProtKB protein identifiers to provide a list of pathways for each protein. The cross-referenced Reactome pathways provide more complete information for each protein than the pathway annotation provided by UniProtKB directly.

4.1.3 OMIM

Online Mendelian Inheritance in Man (OMIM) database [19] is utilized in UniProtKB to provide disease/phenotype information for disease annotations associated with proteins. OMIM is a comprehensive, authoritative and timely knowledgebase of human genes and genetic disorders. Each OMIM entry has a full-text summary of a genetically determined phenotype. UniProtKB carefully links the OMIM entry with the protein entry and describes the natural variant(s) of the protein sequence potentially associated with disease according to the research literature.

4.2 Data Extraction

The data utilized in our experiments is a subset of triples from the UniProtKB RDF dataset. We focus on only human proteins that currently have active entries. The UniProtKB raw data, downloaded from the UniProt website on 4-10-2015, contains about 150 million triples from 971,583 (both reviewed and unreviewed) protein entries. Note that different protein isoforms are represented as different protein entries in UniProtKB. In order to provide a more concise RDF graph to support efficient signature-based indexes, only protein entries associated with at least one of the following statements are extracted: disease annotation, function annotation, PTM annotation, cofactor annotation, subunit annotation and protein interaction. For each protein entry,

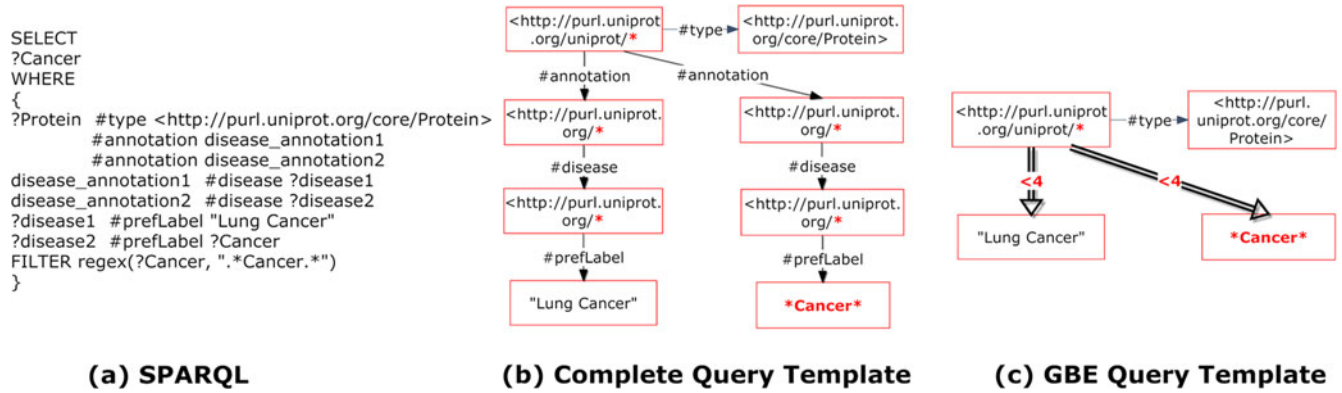


Fig. 8. Query specifications of Q3.

the following properties are extracted: protein names (mnemonic name, recommended name and alternate name), protein organisms (including the taxonomy information), protein keywords, protein tissues, protein gene information (including different gene labels), and protein pathway information (Reactome pathway associated with the protein). The extracted RDF graph contains 89,915 proteins (including different protein isoforms), 4,211 diseases, 1,278 pathways, 18,243 interactions and 35,063 annotations. The size of the extracted RDF graph is about 11.6 million triples including 9 million triples from the taxonomy data. Data extraction can be systematically done from any version of the UniProtKB RDF graph, and more types of annotations can be extracted by changing the specification of the data extraction process.

5 Experimental Results

The proposed framework is implemented with Visual C# 2010 and SQL Server 2008. All experiments were performed on a 2.93 GHZ Intel(R) Xeon machine with 48 GB RAM running Windows Server 2008 R2. The estimated space needed for the NB index is $O(N(\mu/2)^{d_{max}}/m)$, where N is the number of vertices in G, μ is the average node degree, d_{max} is the maximum number of hops used to define the neighbors to be indexed, and m is the binning factor. NB index with a larger d_{max} value results in higher pruning power and ability to handle connection edges with large distance constraints at the cost of requiring more storage space. By using the ByteMap index to hash the RDF labels into IDs, the 2 hop NI index achieves a similar size as the original RDF graph while 3 hop NI index is 8 times larger. As we explain in Section 3.2, a 2-hop NI index can efficiently evaluate connection edges with distance constraints up to 4 hops. Here, we use 2-hop a NI index since all the queries we test involve less than four hops.

In this section, we investigate the ease of utilization of GBE query templates to specify integrated queries. We also assess the strength of these queries in discovering interesting patterns across the integrated network. For this purpose, we focus on 10 queries that require integrated querying across multiple interaction and/or association datasets. We categorize these 10 queries into two groups: 1) single protein patterns: querying the relationships among one protein and other types of resources; 2) multiple protein patterns: querying protein-protein joining based on protein-protein interactions, shared pathways, shared diseases, or shared function. Rather than displaying the results in tabular form, we export query results into Cytoscape [38] to produce meaningfully summarized graphs. For each query, the query results can produce multiple summarized graphs by specifying different relationships among various types of resources.

5.1 Single Protein Patterns

Single protein patterns focus on querying relationships between one protein and other types of entities across different datasets. We explore four queries as single protein patterns:

- Q1. Finding pathways that contain at least one protein associated with "Lung Cancer".
- Q2. Finding molecular functions that are associated with at least one protein associated with "Lung Cancer".
- Q3. Finding cancers that are associated with at least one protein that is also associated with "Lung Cancer". (Motivated by Fig. 1a)
- Q4. Finding tissues that are associated with at least one protein that is associated with "Lung Cancer".

To evaluate these four queries, at least two types of biological resources/entities need to be integrated: protein resource (Uniprot) and disease resource (OMIM). UniProtKB has already linked protein resource with disease information which makes some of queries solvable through manual effort, e.g., one can search all protein entries associated "Lung Cancer" on Uniprot website and manually check all these entries to find any other cancers associated. As UniProtKB also provides a beta SPARQL endpoint which allows users to specify SPARQL queries, some of these queries can also be specified in SPARQL. However, both manual check and specification of complete SPARQL queries require significantly more effort compared to using GBE query templates.

In Fig. 8, we illustrate three possible ways to specify Q3. One can easily observe that both alternatives of specifying the complete query template (Fig. 8b) and the SPARQL query (Fig. 8a) need a good understanding of databases and querying, as well as a good understanding of the underlying graph structure of the UniProt RDF graph. In comparison, by using connection edges and partial keywords, GBE query templates can be much simpler to formulate. As shown in Fig. 8c, the GBE query template uses connection edges to avoid specifying the intermediate annotation node and the disease ID node in identifying the relationship between a protein and a disease. Note that, using connection edges may yield more results compared to specifying all edges in the complete graph template. However, for all these four

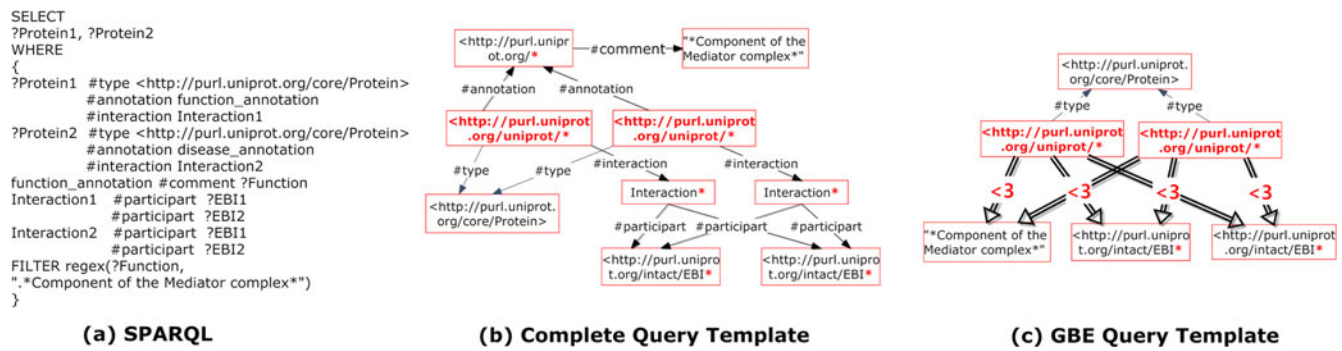


Fig. 9. Query specifications of Q7.

queries, the simplified GBE query templates return the same set of results as the complete graph template on the extracted UniProtKB graph. The GBE framework also supports displaying the results in graph templates. Thus, the researchers can choose a subset of the results that are of interest, and display them as graph templates to perform exploratory search. Due to space constraints, we here do not discuss the details of using the GBE framework to display query results as templates to perform exploratory search.

Q1 returns 39 Reactome pathways potentially associated with "Lung Cancer". Q2 returns 8 functional annotations associated with proteins that are associated with "Lung Cancer". Q3 returns 5 cancers that have overlapping molecular bases with "Lung Cancer". Q4 returns 19 tissues related to proteins associated with "Lung Cancer". As we explain in Section 2, identification of diseases with overlapping molecular bases may be useful in identifying unknown disease-disease relationships. Here, we display the results of Q3 in Fig. 10 using Cytoscape. In Fig. 10a, we display the connections among proteins and diseases identified in the query results of Q3 by making the disease and the protein in each query result as the source and target in Cytoscape. To further provide more clear relationships between "Lung Cancer" and other cancers, another summarized graph is produced in Fig. 10b by representing indirect relationships through proteins as direct relationships. The width of the edge between two diseases is determined by the number of proteins shared by that disease pair. The size of the cancer node is based on the number of shared proteins. The more proteins shared between one disease and "Lung Cancer", the larger the disease node is (same for edge width, the more shared proteins, the wider the edge is). Such networks are utilized in previous studies [7], but such studies require bulk data downloads and significant processing to integrate different datasets. In contrast, this network is constructed via a single query by our GBE-based framework.

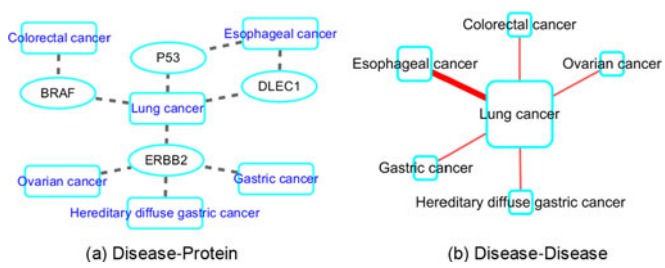


Fig. 10. Query result visualization of Q3.

5.2 Multiple Protein Patterns

Compared to single protein patterns, multiple protein patterns are more complex for query specification and evaluation. We consider three types of protein-protein joining conditions: protein-protein interactions, shared functions, and shared pathways. Some of the multiple protein pattern queries also require joining proteins based on multiple criteria.

5.2.1 *Protein-Protein Interaction*

Protein-protein interactions (PPIs) reveal functional relationships between genes. Although a large number of protein–protein interaction databases are available, there is no direct way to query patterns of protein-protein interactions related with other types of biological entities, e.g., certain phenotypes. Here, we demonstrate the potential power of seamlessly querying PPI data along with other sources of data. For this purpose, we consider two queries:

- Q5. Finding cancers associated with at least one protein that interacts with another protein associated with "Breast Cancer". (Motivated by Fig. 1b)
- Q6. Finding all pairs of cancers associated with pairs of proteins that interact with each other. (Motivated by Fig. 1b)

As illustrated in Section 2, Fig. 1b, the motivation for these two queries is to find shared molecular interactions between diseases. Q6 is a more general form of Q5 as Q6 tries to identify all pairs of cancers with shared molecular interactions while Q5 specifies one disease as "Breast Cancer". Complete visualization of query results for Q5 is shown in Fig. 12. The width of the edge between two proteins (represents the binary interaction between two proteins) is proportional to the number of disease pairs that share this interaction. Four protein-protein interaction patterns are identified that connect different cancers to "Breast cancer". Compared to Fig. 10a, one can observe that there is no edge between BRCA1 protein and "Breast Cancer" in Fig. 12. This is a result of graph isomorphism matching of the query template as each query node should match a unique node in the RDF graph. Since there is no protein associated with other cancers which interacts with BRCA1 protein, no query results contain an edge between BRCA1 protein and "Breast Cancer" for Q5. The summarized disease-disease relationships identified by Q5 is shown in Fig. 11. Similar to Fig. 10b, the edge width represents the number of protein-protein interactions shared by the respective disease pair. For Q6, the complete disease-disease relationships among all pairs of cancers is shown in Fig. 13.

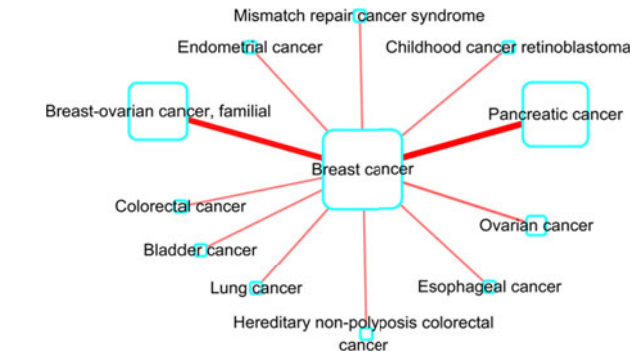


Fig. 11. Disease-disease relationships for Q5.

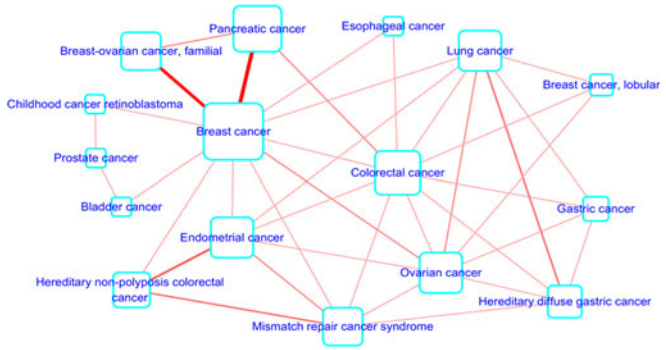


Fig. 13. Disease-disease relationships for Q6.

5.2.2 Shared Functions

Identifying network schemas requires protein-protein joining through both shared functions and protein-protein interactions as shown. The general form of these types of queries is shown in Section 2, Fig. 1c. Here, we consider one query to evaluate shared functions:

Q7. Finding pairs of proteins that have function description as "Component of the Mediator complex" and interact with each other. (Motivated by Fig. 1c)

Specifying queries of multiple protein patterns is more difficult compared to queries involving single protein patterns. Similar to Fig. 8 of Q3, all three possible ways (SPARQL, complete graph template and simplified GBE template) to specify Q5 are shown in Fig. 9. Note that the interaction between each pair of proteins are identified by at least two observations (IntAct EBI nodes), as required by UniprotKB. The query results are displayed in Fig. 14. Three groups of proteins are identified where proteins in the same group densely interact with each other.

5.2.3 Shared Pathways

Having a better understanding of the roles that proteins play at higher order interconnected pathways is critical to understanding molecular reactions at cellular level. Some conceptualizations of protein-protein interactions also consider pro-teins in the same pathway interact with each other directly or indirectly. Identifying protein-protein patterns with shared pathways provides complementary information to binary protein-protein interactions. To illustrate the power of GBE-based querying in integrating PPI and pathway associations, we consider the following three queries:

Q8. Finding any pair of proteins that are involved in the same pathway and are both associated with "Breast Cancer".

Q9. Finding cancers associated with a protein that is involved in the same pathway with a protein associated with "Breast Cancer". (Motivated by Fig. 1e)

Q10. Finding two proteins that are involved in the same pathway and interact with each other such that one protein is associated with "Breast Cancer" and the other is associated with another cancer.

The query results of Q8 are visualized in Fig. 15. We find that there are two groups of proteins where the proteins in each group are closely related with each other through shared pathways. Four proteins associated with three cancers that interact with each other and share three common pathways are identified in Q10 as shown in Fig. 16. Note that "Breast Cancer", "Pancreatic Cancer", and "Breast-Ovarian Cancer" are also noticed to share molecular interactions in Figs. 11 and 13. Q9 returns 24 distinct proteins, 31 pathways and 15 cancers (not including "Breast Cancer").

5.3 Index Space Comparison

As explained in Section 3.1, neighborhood byte (NB) index is more space efficient as compared to neighborhood integer (NI) index proposed in [35]. The significant byte, which is utilized to group the neighbor IDs into one index entry saves considerable space if the indexed node has many neighbors sharing the same significant byte. Indicated by the flag bit, all neighbor IDs indexed are represented as the byte arrays with fixed length (two bytes if the flag bit is 0 and three bytes if the flag bit is 1). The fixed length byte IDs

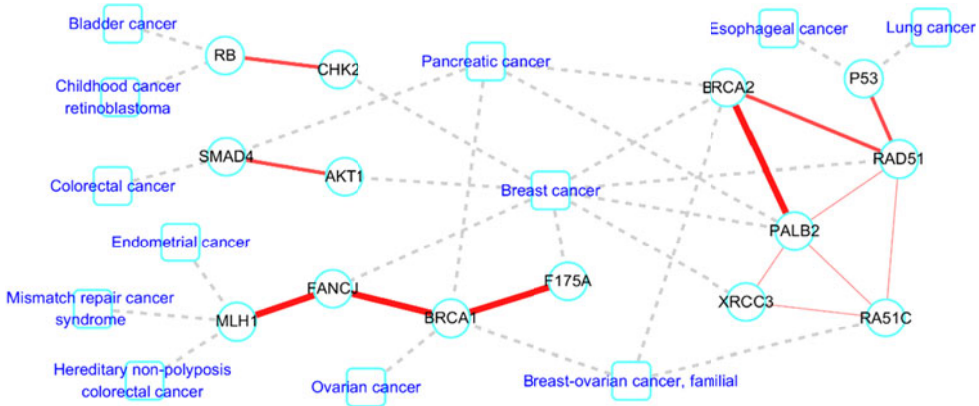


Fig. 12. Full visualization of query results for Q5.

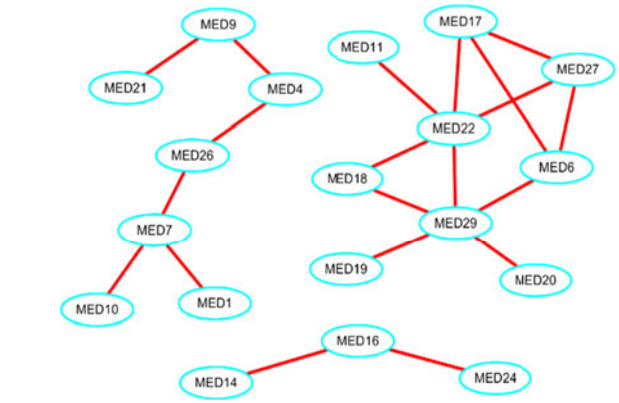


Fig. 14. Protein-protein relationships for Q7.

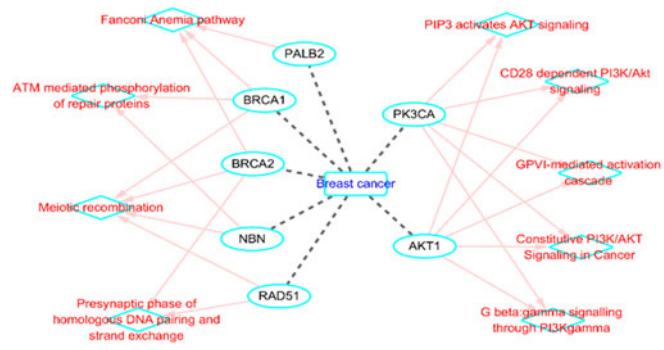


Fig. 15. Protein-protein relationships for Q8.

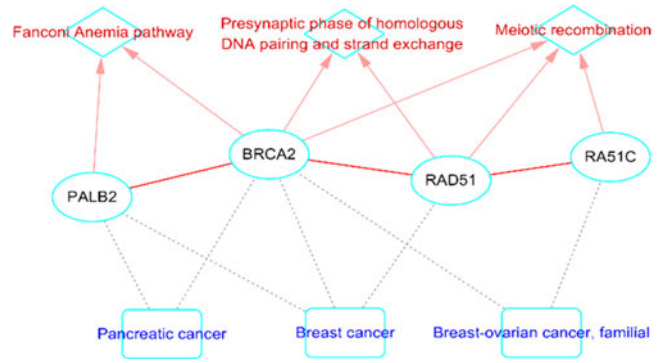


Fig. 16. Protein-protein relationships for Q10.

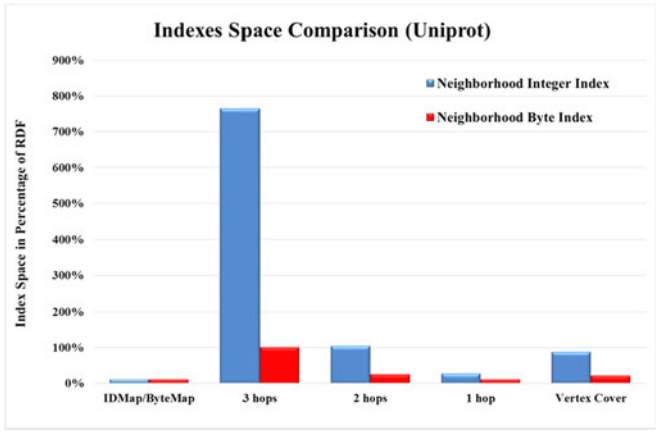


Fig. 17. Index space comparison (Uniprot).

allow the query framework to separate each neighbor ID from a long byte array without using any delimiter. The index space comparison results are shown in Fig. 17 (for Uniprot dataset). One can notice that the IDMap index and the ByteMap index are equivalent in space since the IDs in the IDMap index are 32-bit integers while the IDs in the ByteMap index are 4 byte bit arrays. Three versions of the neighborhood indexes are compared: 3-hop neighborhood index, 2-hop neighborhood index and vertex cover neighborhood index (the vertex cover neighborhood index uses 2-hop neighbors for nodes in the vertex cover set, and 1 hop neighbors for all other nodes). As seen in Fig. 17, the neighborhood byte index achieves higher compression rate if more neighbors are indexed, since more neighbors share their most significant bytes. Overall, the neighborhood byte index uses less than half of the index space compared to the neighborhood integer index for both datasets in all versions.

5.4 Query Evaluation Efficiency

Neighborhood signature index is considered as a strong candidate to evaluate GBE templates as it can: 1) reduce the unnecessary candidates in subgraph isomorphism matching, and 2) handle connection edges. We re-evaluate the ten queries on the Uniprot dataset to assess the efficiency of utilizing neighborhood byte index in RDF-h. All these queries are proposed as small components connected by connection edges and the query performance highly depends on the efficiency of the connectivity check algorithm. Query runtime results are shown in Table 1.

We find that single protein pattern queries are more efficient to evaluate than multiple protein pattern queries. Among single protein queries, Q2 requires the longest running time as the component of matching any protein associated with functional annotations results in a large number of matching candidates, which leads to a large number of connection checks. Q6 is the most time consuming multiple protein pattern query, as each decomposed component is small and all components are connected through connection edges. Compared to Q6, Q5 requires less effort as one of the cancers is specified as "Breast cancer". Specification of the label of one of the nodes significantly reduces the number of intermediate matches. Except for Q6, all queries can be evaluated in less than twenty seconds, can be considered real-time.

Comparison of the query performance using NB index and Merge Check with the query performance using NI index and Interval Check is shown in Fig. 18. We find that Merge Check algorithm is much more efficient to handle connection edges since it enables early skip by checking the significant byte in neighborhood byte index, which saves more than 50 percent of the runtime for most of the queries.

6 Related Work

Many of the existing RDF systems, including Sesame [39], Virtuoso [40], gStore [41] and Jena [42], use SPARQL [43] as the default query language. SPARQL is a query language that uses graph patterns as basic query units and evaluates queries via triple joins. Currently, SPARQL supports path queries defined with regular expressions (property paths). However, most of these systems have difficulties in processing the graph-by-example (GBE) templates similar to the

TABLE 1
Query Running Times for All Queries

Query ID	Q1	Q2	Q3	Q4	Q5
# of Results	72	13	10	48	25
Run Time (ms)	719	7,618	561	784	16,448
Query ID	Q6	Q7	Q8	Q9	Q10
# of Results	67	20	32	158	8
Run Time (ms)	39,987	3,422	8,421	8,085	9,495

examples shown in Figs. 8c and 9c for the following reasons: 1) it is difficult to identify and join connection edges without using specific indexes; 2) there is no well-defined mechanism to address partial labels; 3) due to the exponential space of subgraphs, query templates may generate too many intermediate candidates if no pruning techniques are utilized.

To tackle the challenges associated with the subgraph isomorphism problem, signature based indexes are commonly utilized [44], [45], [46], [47], [48], [49]. TALE [44] and SAPPER [45] index the neighbors of each node by hashing into bit arrays and utilizing these bit arrays as signatures to check neighborhood containment. In contrast, graphQL [46] and SPath [47] use neighborhood indexes directly to index node labels. In GraphQL, the neighbors of each node are indexed as a sequence of node labels in lexicographic order. gStore [48], [49] extends the utilization of signature-based pruning to RDF datasets. The neighborhood signature is proposed as a bit string according to the adjacent edge labels and node labels. All vertex neighborhood signatures are then indexed using a special index schema, VS tree, to provide efficient query evaluation.

Storing biolmedical data in RDF is a current trend. In addition to the UniProt dataset utilized in our experiments, many other biolomedical datasets are available in RDF format from the Bio2RDF project [50]. The databases that provide data in RDF format include chEMBL, ClinicalTrials, DrugBank, iProClass, KEGG, MeSH.

7 Conclusion

This paper has demonstrated that graph template matching based querying, which applies simple graph templates to query an integrated RDF knowledge base, is a promising tool to express sophisticated questions, and discover hidden connections among biological or biomedical entities from diverse datasets. RDF is being adopted increasingly to publish or exchange data in the web, and vast amounts of RDF data are already available. The availability of ever increasing amounts of RDF data in various fields of biomedical applications from public and private sources, adds even more to the potential of querying integrated RDF data sets for new insights and discovering knowledge about hidden associations among biological and medical entities.

Future work includes building a web-based graphical user interface to enable biomedical researchers to use this framework for exploratory querying. Using more diverse and much larger collections of RDF data, including data on publications, sequences, and drugs as data sets, will enable us to query even more sophisticated queries by directly using graph template querying. Finally, we are currently working on the distributed version of our querying framework and combining our work with comprehensive genomics data.

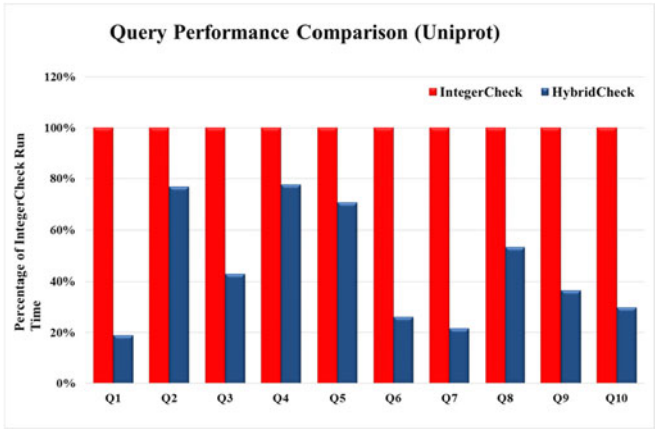
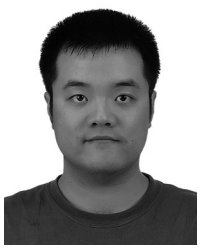


Fig. 18. Query peformance comparison.

References

[1] R. Mosca, T. Pons, A. Céol, A. Valencia, and P. Aloy, "Towards a detailed atlas of protein–protein interactions," *Current Opinion Structural Biol.*, vol. 31, no. 6, pp. 929–940, 2013.
[2] S. L. Carter, C. M. Brechbühler, M. Griffin, and A. T. Bond, "Gene co-expression network topology provides a framework for molecular characterization of cellular state," *Bioinf.*, vol. 20, pp. 2242–2250, 2004.
[3] Z. Hu, P. J. Killion, and V. R. Iyer, "Genetic reconstruction of a functional transcriptional regulatory network," *Nature Genetics*, vol. 39, no. 5, pp. 683–687, 2007.
[4] A. R. Pah, R. Guimera, A. M. Mustoe, and L. A. N. Amaral, "Use of a global metabolic network to curate organismal metabolic networks," *Scientific Rep.*, vol. 3, 2013, Art. no. 1695.
[5] A. H. Y. Tong, et al., "Global mapping of the yeast genetic interaction network," *Science*, vol. 303, no. 5659, pp. 808–813, 2004.
[6] A. Ritz, A. N. Tegge, H. Kim, C. L. Poire, T. M. Muraliemail "Signaling hypergraphs," *Trends Biotechnology*, vol. 32, no. 7, pp. 356–362, 2014.
[7] K. I. Goh, M. E. Cusick, D. Valle, B. Childs, M. Vidal, and A. L. Barabási, "The human disease network," *Proc. Nat. Academy Sci. United States America*, vol. 104, no. 21, pp. 8685–8690, 2007.
[8] J. Von Eichborn, M. S. Murgueitio, M. Dunkel, S. Koerner, P. E. Bourne, and R. Preissner, "PROMISCUOUS: A database for network-based drug-repositioning," *Nucleic Acids Res.*, vol. 39, no. suppl 1, pp. D1060–D1066, 2011.
[9] I. Vogt and J. Mestres, "Drug-target networks," *Molecular Informat.*, vol. 29, no. 1/2, pp. 10–14, 2010.
[10] A. Lisewski, J. P. Quiros, C. Ng, A. Adikesavan, and K. Miura, "Supergenomic network compression and the discovery of EXP1 as a glutathione transferase inhibited by artesunate," *Cell*, vol. 158, pp. 916–928, 2014.
[11] M. Taşan, G. Musso, T. Hao, M. Vidal, C. MacRae, and F. Roth, "Selecting causal genes from genome-wide association studies via functionally coherent subnetworks," *Nature Methods*, vol. 12, pp. 154–159, 2015.
[12] Y. Li and J. C. Patra, "Genome-wide inferring gene–phenotype relationship by walking on the heterogeneous network," *Bioinf.*, vol. 26, no. 9, pp. 1219–1224, 2010.
[13] F. Cheng, et al., "Prediction of drug-target interactions and drug repositioning via network-based inference," *PLoS Comput. Biol.*, vol. 8, no. 5, 2012, Art. no. e1002503.
[14] M. Cao, C. M. Pietras, X. Feng, and K. J. Doroschak, "New directions for diffusion-based network prediction of protein function: Incorporating pathways with confidence," *Bioinf.*, vol. 30, no. 12, , pp. i219–i227, 2014.
[15] R. Cyganiak, D. Wood, and M. Lanthaler. "RDF 1.1 concepts and abstract syntax," *W3C Recommendation*, vol. 25, pp. 1–8, 2014.
[16] UniProt Consortium, "UniProt: A hub for protein information," *Nucleic Acids Res.*, vol. 43, no. D204–D212, 2015.
[17] B. Aranda, et al., "The IntAct molecular interaction database in 2010," *Nucleic Acids Res.*, vol. 38, pp. D525–D531, 2010.

[18] D. Croft, et al., "The Reactome pathway knowledgebase," *Nucleic Acids Res.*, vol. 42, pp. D472–D477, 2014.
[19] A. Hamosh, A. F. Scott, J. S. Amberger, and C. A. Bocchini, "Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders," *Nucleic Acids Res.*, vol. 33, pp. D514–D517, 2005.
[20] A. Subramanian, et al., "Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles," *Proc. Nat. Academy Sci. United States America*, vol. 102, no. 43, pp. 15545–15550, 2005.
[21] R. K. Nibbe, M. Koyutürk, and M. R. Chance, "An integrative-omics approach to identify functional sub-networks in human colorectal cancer," *PLoS Comput. Biol.*, vol. 6, no. 1, 2010, Art. no. e1000639.
[22] K. L. McGary, T. J. Park, J. O. Woods, H. J. Cha, J. B. Wallingford, and E. M. Marcotte, "Systematic discovery of nonobvious human disease models through orthologous phenotypes," *Proc. Nat. Academy Sci. United States America*, vol. 107, no. 14, pp. 6544–6549, 2010.
[23] J. O. Woods, U. M. Singh-Blom, J. M. Laurent, K. L. McGary, and E. M. Marcotte, "Prediction of gene–phenotype associations in humans, mice, and plants using phenologs," *BMC Bioinf.*, vol. 14, no. 1, 2013, Art. no. 203.
[24] Y. Liu, M. Koyutürk, S. Maxwell, Z. Zhao, and M. R. Chance, "Integrative analysis of common neurodegenerative diseases using gene association, interaction networks and mRNA expression data," in *Proc. AMIA Summits Translational Sci. Proc.*, 2012, Art. no. 62.
[25] J. Menche, et al., "Uncovering disease-disease relationships through the incomplete interactome," *Science*, vol. 347, no. 6224, pp. 1257601-1–1257601-8, 2015.
[26] J. Li, S. Zheng, B. Chen, A.J. Butte, S. J. Swamidass, and Z. Lu, "A survey of current trends in computational drug repositioning," *Briefings Bioinf.*, vol. 17, pp. 2–12, 2015.
[27] E. Banks, E. Nabieva, R. Peterson, and M. Singh, "NetGrep: Fast network schema searches in interactomes," *Genome Biol.*, vol. 9, no. 9, 2008, Art. no. R138.
[28] J. Pandey, M. Koyutürk, Y. Kim, W. Szpankowski, S. Subramaniam, and A. Grama, "Functional annotation of regulatory pathways," *Bioinf.*, vol. 23, 13, pp. i377–i386, 2007.
[29] M. Koyuturk, "Using protein interaction networks to understand complex diseases," *Computer*, vol. 45, no. 3, pp. 31–38, 2012.
[30] A. Baryshnikova, et al., "Synthetic genetic array (SGA) analysis in Saccharomyces cerevisiae and Schizosaccharomyces pombe," *Methods Enzymology*, vol. 470, pp. 145–179, 2010.
[31] A. Vinayagam, et al., "Integrating protein-protein interaction networks with phenotypes reveals signs of interactions," *Nature Methods*, vol. 11, no. 1, pp. 94–99, 2014.
[32] J. Falck, J. H. Petrini, B. R. Williams, J. Lukas, and J. Bartek, "The DNA damage-dependent intra–S phase checkpoint is regulated by parallel pathways," *Nature Genetics*, vol. 30, no. 3, pp. 290–294, 2002.
[33] M. Bailly-Bechet, et al., "Finding undetected protein associations in cell signaling by belief propagation," *Proc. Nat. Academy Sci. United States America*, vol. 108, no. 2, pp. 882–887, 2011.
[34] P. Shannon, et al., "Cytoscape: A software environment for integrated models of biomolecular interaction networks," *Genome Res.*, vol. 13, no. 11, pp. 2498–2504, 2003.
[35] S. Qiao, M. Koyutürk, and M. Özsoyoğlu, "Integrated querying of disparate association and interaction data in biomedical applications," in *Proc. 6th ACM Conf. BCB*, 2015, pp. 146–155.
[36] Y. Li and J. M. Patel, "BitWeaving: Fast scans for main memory data processing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2013, pp. 289–300.
[37] Z. Sun, H. Wang, H. Wang, B. Shao, and J. Li, "Efficient subgraph matching on billion node graphs," *Proc. VLDB Endowment*, vol. 5, no. 9, pp. 788–799, 2012.
[38] C. T. Lopes, M. Franz, F. Kazi, S. L. Donaldson, Q. Morris, and G. D. Bader, "Cytoscape Web: an interactive web-based network browser," *Bioinformatics*, vol. 26, no. 18, pp. 2347–2348, 2010.
[39] J. Broekstra, A. Kampman, and F. Van Harmelen, "Sesame: A generic architecture for storing and querying RDF and RDF schema," *Semantic Web—ISWC*, vol. 2342, p. 54–68, 2002.
[40] O. Erling and I. Mikhailov, "RDF support in the Virtuoso DBMS," in *Networked Knowledge - Networked Media*. Berlin, Germany: Springer, 2007.
[41] S. Harris and N. Gibbins, "3store: Efficient bulk RDF storage," in *Proc. PSSS*, 2003, pp. 1–15.
[42] K. Wilkinson, C. Sayers, H. A. Kuno, and D. Reynolds, "Efficient RDF storage and retrieval in Jena2," in *Proc. 1st Int. Conf. Semantic Web Databases*, 2003, pp. 131–150.
[43] J. Pérez, M. Arenas, and C. Gutierrez, "Semantics and complexity of SPARQL," *ACM Trans. Database Syst.*, vol. 34, no. 3, 2009, Art. no. 16.
[44] Y. Tian and J. M. Patel, "Tale: A tool for approximate large graph matching," in *Proc. IEEE 24th Conf. ICDE*, 2008, pp. 963–972.
[45] S. Zhang, J. Yang, and W. Jin, "Sapper: Subgraph indexing and approximate matching in large graphs," *Proc. VLDB Endowment*, vol. 3, no. 1/2, pp. 1185–1194, 2010.
[46] H. He, and A. Singh, "Graphs-at-a-time: query language and access methods for graph databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2008, pp. 405–418.
[47] P. Zhao and J. Han, "On graph query optimization in large networks," *Proc. VLDB Endowment*, vol. 3, no. 1/2, pp. 340–351, 2010.
[48] L. Zou, M. T. Özsu, L. Chen, X. Shen, R. Huang, and D. Zhao, "gStore: A graph-based SPARQL query engine," *VLDB J.*, vol. 23, no. 4, pp. 565–590, 2014.
[49] L. Zou, J. Mo, L. Chen, M. T. Özsu, and D. Zhao, "gStore: Answering SPARQL queries via subgraph matching," *Proc. VLDB Endowment*, vol. 4, no. 8, pp. 482–493, 2011.
[50] F. Belleau, M. A. Nolin, N. Tourigny, P. Rigault, and J. Morissette, "Bio2RDF: Towards a mashup to build bioinformatics knowledge systems," *J. Biomed. Inform.*, vol. 41, no. 5, pp. 706–716, 2008.



Shi Qiao received the BS degree from Nanjing University and the PhD degree in computer science from Case Western Reserve University. He is currently a software development engineer at Microsoft Corporation. His research has been published in leading academic conferences including SIGMOD, VLDB, BCB, etc., and he received the Best Student Paper Award from ACM-BCB'2015. His research interests include query processing and optimization, RDF, and bioinformatics.

Mehmet Koyutürk received the BS and MS degrees from Bilkent University, respectively, in electrical engineering and computer engineering and the PhD degree in computer science from Purdue University. He is currently T. & A. Schroeder associate professor in the Department of Electrical Engineering and Computer Science, Case Western Reserve University. His research focuses on the analysis of biological networks, systems biology of complex diseases, and computational genomics. He serves as an associate editor for the *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* and the *EURASIP Journal on Bioinformatics and Systems Biology*.

Meral Z. Özsoyoğlu received the BSc and MSc degrees are from Middle East Technical University in elecetrical engineering and computer science, respectively, and the PhD degree in computer science from the University of Alberta. She is currently Jennings professor of computer science in the Department of Electrical Engineering and Computer Science at Case Western Reserve University. Her primary work and research interests include the areas of query languages and query processing, data models, and index structures in databases, including scientific databases, bio-informatics, and medical informatics. She has also served in a variety of leadership roles in the computer science research community, including the program chair of conferences VLDB 2012, IEEE ICDE 2004, ACM PODS 1997 and SSDBM 1999, and the editor-in-chief of ACM TODS 2007-2014 and PVLDB, 2011-2012. She has been a trustee of the VLDB Endowment, an associate editor of the *ACM TODS*, and the *IEEE TKDE* and vice chair of ACM SIGMOD. She is an ACM fellow, recipient of the IBM Faculty Award, NSF Faculty Award for Women, and a Distinguished Alumni Award from University of Alberta.