Building an online UMLS knowledge discovery platform using graph indexing

THESIS

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Aaron David Albin

Graduate Program in Computer Science and Engineering

The Ohio State University

2014

Master's Examination Committee:

Dr. Yang Xiang, Advisor

Dr. Rajiv Ramnath
Abstract

The UMLS is a rich collection of biomedical concepts which are connected by semantic relations. Using transitively associated information for knowledge discovery has been shown to be effective for many applications in the biomedical field. Although there are a few tools and methods available for extracting transitive knowledge from the UMLS, they usually have major restrictions on the length of transitive relations or on the number of data sources.

To overcome these restrictions, the web platform onGrid was developed to support efficient path queries and knowledge discovery on the UMLS. This platform provides several features such as converting natural language queries into UMLS concepts, performing efficient queries, and visualizing the result paths. It also builds relationship and distance matrices for two sets of biomedical terms, making it possible to perform effective knowledge discovery on these concepts.

onGrid can be applied to study biomedical concept relations between any two sets or within one set of biomedical concepts. In this work, onGrid is used to study the gene-gene relationships in HUGO as well as disease-disease relationships in OMIM. By cross validating the results with external datasets, it is demonstrated that onGrid is very efficient to be used for conceptual-based knowledge discovery on the UMLS.

onGrid is a very efficient tool for querying the UMLS for transitive relations, studying relationships between biomedical terms, and generating hypotheses. The online UMLS
knowledge discovery platform has been tested on the BMI Netlab server (URL: https://netlab.bmi.osumc.edu/ongrid).
Dedication

This document is dedicated to my family, whose continual encouragement enabled the completion of my studies, and my dear friend Qian Zhang for her unwavering support.
Acknowledgments

I would like to thank Dr. Yang Xiang for his diligent guidance as my advisor throughout my graduate study, Dr. Rajiv Ramnath for his time serving on my graduate committee, and Xiaonan Ji for her assistance while developing the online portion of this project.
Vita

May 2011 ..............................................B.S. Computer Science, Southeastern Louisiana University

May 2014 ..............................................M.S. Computer Science & Engineering, The Ohio State University

Fields of Study

Major Field: Computer Science and Engineering
Table of Contents

Abstract .................................................................................................................................................. ii
Dedication .............................................................................................................................................. iv
Acknowledgments ................................................................................................................................. v
Vita .......................................................................................................................................................... vi
List of Tables .......................................................................................................................................... ix
List of Figures .......................................................................................................................................... x

Chapter 1: Introduction .......................................................................................................................... 1
  1.1 Related Work .................................................................................................................................. 2
  1.2 Thesis Organization ....................................................................................................................... 6

Chapter 2: Indexing the UMLS .............................................................................................................. 7
  2.1 Algorithm Framework ................................................................................................................... 8
  2.2 Indexing Time Complexity Analysis ............................................................................................... 10

Chapter 3: System Framework for Online Queries Using Indexing ....................................................... 11
  3.1 The Client-Server Framework ....................................................................................................... 11
  3.2 Handling Concurrent Requests ..................................................................................................... 12
  3.3 Query Types ................................................................................................................................... 13
Chapter 4: onGrid Knowledge Discovery ................................................................. 19

4.1 Disease-Gene ........................................................................................................ 19

4.2 Gene-Gene ............................................................................................................ 27

Chapter 5: Conclusion ............................................................................................... 29

Bibliography ................................................................................................................ 30
List of Tables

Table 1. Corresponding thresholds $\delta$ for kDLS and onGrid........................................ 21

Table 2. Top 5 gene clusters and their enrichments using ToppGene .............................. 27
List of Figures

Figure 1. Selected subgraph from the UMLS Semantic Network ........................................ 8
Figure 2. Flowchart of the onGrid Framework................................................................. 12
Figure 3. An illustration of visualized paths between two UMLS concepts ..................... 15
Figure 4. An example of a relationship and distance matrix generated by onGrid .......... 17
Figure 5. Max Folder Enrichment $f_{\alpha}$ at thresholds $\delta$ ranging from 0.45 to 1.4 ........ 20
Figure 6. Folder Enrichment $f_{\alpha}$ at thresholds $\delta$ for both kDLS and onGrid ............. 20
Figure 7. Top diseases related to Adenocarcinoma of lung by on Grid ($\delta=1.1$) ........ 23
Figure 8. Top diseases related to Adenocarcinoma of lung according to CTD .............. 24
Figure 9. Top diseases related to Glioblastoma by onGrid ($\delta = 1.1$) ......................... 25
Figure 10. Top diseases related to Glioblastoma according to CTD ............................. 26
Chapter 1: Introduction

In biomedical science, the Unified Medical Language System (UMLS) [1] is the largest thesaurus widely used in various applications. It is a collection of more than 100 controlled vocabularies and consists of three parts: the Metathesaurus, Semantic network, and Specialist Lexicon. The Metathesaurus comprises the majority of information in the UMLS and has over two million concepts, each with a Concept Unique Identifier (CUI), and over 15 million links (associations) between pairs of CUIs.

Given the amount of information contained within the UMLS, there is much potential for applying modern methods of knowledge discovery. The UMLS can fundamentally be viewed as a source of many transitively related concepts. It will be necessary to demonstrate why this information is useful in novel hypothesis generation. Furthermore, given the large size of the UMLS, we must determine how to approach the task efficiently. The goal of this work, then, is to build upon recent knowledge discovery techniques for the UMLS and develop an accessible tool for the biomedical community.

The result is an online knowledge discovery web platform UMLS-onGrid. Utilizing a k-neighborhood Decentralization Labeling Scheme (kDLS) to index the UMLS graph [2], onGrid allows for shortest path visualization between two biomedical concepts as well as the generation of distance and enhanced relationship matrices between two sets of concepts. Data mining and cross-validation is performed on disease-gene relationships and gene-disease relationships in OMIM and HUGO as case studies to demonstrate the
effectiveness of onGrid for conceptual-based knowledge discovery. In addition, the results for the disease-gene study generated from UMLS-onGrid are compared with those from the original kDLS algorithm, demonstrating the improvements made with UMLS-onGrid. The results in this thesis were partially presented in [3].

1.1 Related Work

Transitive associations have been important sources for hypothesis generation in biomedical science since Swanson's discovery of the connection between fish oil and Raynaud's syndrome via blood viscosity [4]. In Swanson’s paradigm, an association between concepts A and C may be possible if both are related to a third concept B. A number of discoveries and hypotheses have been made under this model. For instance, Hristovski et al. proposed literature-based discovery to search disease candidate genes [5], to investigate drug mechanisms [6], and to identify novel therapeutic approaches [7]. As another example, Petric et al. used this model to study autism by mining literature, and found the connection between autism and calcineurin [8].

Transitive Knowledge Discovery using the UMLS

With the UMLS, such transitive association studies could be powerful in generating novel hypotheses. Various efforts have been made to this effect. One example is the interactive biomedical discovery support system (BITOLA) developed by Hristovski et al. [5] [9] which supports the input of UMLS CUIs, concept semantic types, and chromosome locations, in searching for hypothetical relations such as disease candidate
genes. BITOLA is based on Swanson's one transitive relationship model. It is natural to consider if multiple transitive relationships will generate more rich hypotheses. In [10], Wilkowski et al. showed that by extracting paths from a graph modeling concepts and their relations, it is possible to extend the one transitive relationship model to a multiple-transitive relationship model for novel hypothesis discovery. For the UMLS, if we consider each CUI as a vertex, and links connecting two CUIs as an edge, we obtain a graph modeling the UMLS. The transitively-associated queries on the UMLS can be regarded as queries on the UMLS graph. In fact, a number of works [11] [12] [13] [14] [15] [16] have successfully used multiple-transitive relationships in the UMLS to measure the closeness between two concepts.

However, these works have two major limitations. First, similar as [10], they rely on ad-hoc path search algorithms such as depth-first search, which limits their searching ability on very large graphs. Thus they put major limitations on their search ranges, such as within a very small number of data sources in the UMLS, or very short search paths (e.g., no more than 5 concepts in a path in [13]). Second, they generally rely on distance or a shortest path to determine the closeness between two concepts. In this case, a false or weak transitive relationship may nullify the whole hypothesis. Given this observation, we conclude that this is not as reliable as a measurement on a large collection of paths. In fact, the effective measurement of a relationship between two concepts in [5] and [17] can be viewed as a measurement on a collection of very short paths.
Graph indexing for distance queries

To address the first limitation, we consider approaches taken in the graph database community. Graph indexing is a technique designed to efficiently answer reachability or distance queries on large graphs. In particular, the 2-hop cover strategy proposed by Cohen et al [18] has become well-known, and suggests assigning index labels to vertices. For a given vertex, its label contains a record of intermediate vertices that it can reach as well as those that can reach it. A query can be answered by comparing the in-label of one vertex with the out-label of another to determine if an intermediate vertex is shared. However, this approach has been shown to be very limited in terms of scalability. In [19] it is noted that while the 2-hop strategy works well on sparse graphs, it is not very efficient when graphs are dense (that is, having a high ratio of edges to vertices) due to the construction time of the labels. There have been incremental improvements [19], including the use of a divide-and-conquer computation of the 2-hop cover [20] [21], exploiting strongly connected components and partitioning the graph [22], and using a highway to replace the intermediate vertex in the 2-hop strategy (known as highway-centric labeling) [23]. Despite these efforts, it remains a challenge to efficiently answer queries on graphs with similar size and density as the UMLS graph. More recently, [24] proposed a TreeMap approach which can efficiently answer distance queries on many real graphs having a small tree decomposition width. However, it is not clear how to obtain a small tree decomposition width for the UMLS graph.
Thus, by studying the composition of the UMLS itself, Xiang et al. developed a k-neighborhood Decentralization Labeling Scheme (kDLS) to efficiently index the UMLS [2]. kDLS supports efficient path and distance queries on the whole UMLS, as well as a measurement on the closeness between any two UMLS concepts by a collection of paths found between them. kDLS utilizes the power-law property of the UMLS for designing the indexing algorithm, and turns out to be very effective in indexing the UMLS for both answering graph queries and discovering knowledge. Explained briefly, the indexing algorithm of kDLS iteratively removes a high degree vertex from the UMLS graph and broadcasts its information to the remaining vertices in the k neighborhood of the removed vertex. This information becomes the label for each vertex, such that by comparing the labels it is possible to find a collection of paths (including but not limited to shortest paths) between the two vertices. The kDLS method is guaranteed to find at least one shortest path if the two vertices are within k hops on the UMLS graph. On average, the number of paths discovered by kDLS is much larger than by BFS or DFS, as shown in [2]. Subsequently, the measurement between two concepts is based on the number of paths discovered as well as their lengths. kDLS has demonstrated its power in medical concept coreference resolution in clinical text [25].

However, kDLS has several major disadvantages. It does not take into account the semantic networks in the UMLS, does not accept natural language based queries, and the configuration requirement impedes the wide application of kDLS in the biomedical domain. To address these issues, an online UMLS knowledge discovery platform
(UMLS-onGrid) was developed. UMLS-onGrid is very effective for knowledge discovery and hypotheses generation in biomedical science.

1.2 Thesis Organization

In chapter 2, an overview of the kDLS algorithm is given, with a brief justification for why it is well-suited for the UMLS. We then extend kDLS by incorporating information from the UMLS Semantic Network. Algorithm pseudocode is presented with subsequent analysis of the indexing time complexity, the primary hindrance of prior techniques.

In chapter 3, the system framework for the UMLS-onGrid tool is described. It addresses the issues of natural language queries and handling concurrent requests. Additionally, the functionality of onGrid is shown, including shortest-path visualization and large scale relationship matrix generation.

In chapter 4, two cases studies are presented to demonstrate the effectiveness of the UMLS-onGrid system. Cross-validation and analysis is performed on matrices generated for gene-gene and disease-gene relationships. Results for disease-gene relationships are compared with those generated from the original kDLS algorithm in order to compare performance.

Finally, conclusions and discussions on future work are given in chapter 5.
In the kDLS algorithm, the process of removing a vertex and broadcasting its information to vertices in its k neighborhood is called decentralization [2]. Two attributes of the UMLS graph make this a reasonable approach. First, the graph follows a power law such that the larger the degree, the fewer vertices are observed. Second, it exhibits the small-world phenomenon which implies that the majority of paths can be found within a reasonably short distance. Additionally, the indexing construction process can be simplified further by first removing sink and source vertices. That is, respectively, vertices which contain a large number of incoming edges with no outgoing edges, and vertices which contain a large number of outgoing edges with no incoming edges. In these cases, vertices tend to be more abstract concepts and thus are less relevant for knowledge discovery.

This work extends kDLS by incorporating the UMLS Semantic Network to refine the knowledge discovery process. In the UMLS, each vertex is associated with one or more semantic type. There are 133 unique semantic types, organized in a directed acyclic graph. The semantic types closer to the root level are more abstract than those at the leaf level.
From Figure 1, it is simple to observe that biomedical concepts associated with “Entity” would be more abstract than those associated with “Clinical Drug”. To reflect this level of concreteness, the semantic types are given a rank determined by reverse topological ordering. All leaves in the semantic network have reverse topological level 1. After removing these leaves, all leaves in the new network have reverse topological order 2. Iteratively applying this approach determines a reverse topological level for all semantic types. These ranks then become an attribute which is broadcast during the indexing process.

2.1 Algorithm Framework

The decentralization process is done using Breadth-First Search (BFS) to broadcast vertices information to its k-neighborhood. The high level workflow follows that of
kDLS [2]. To facilitate our discussion, we present its pseudocode in Procedure 1 and 2.

1. **Procedure 1 run(G, k):**
   2. create a list A
   3. for all vertices v in G:
      4. if v is a sink or source vertex then
         5. append v to A
   6. for all vertices v in A:
      7. remove v from G
   8. for all vertices v in G:
      9. BFSLabel(G, v, k)
   10. remove v from G

1. **Procedure 2 BFSLabel(G, v, k):**
   2. create a queue Q
   3. initialize v.level to 0, v.score to v.rank
   4. enqueue v onto Q
   5. while Q is not empty loop:
      6. t ← Q.dequeue()
      7. if t.level < k then
         8. for all edges e in G.adjacentEdges(t) loop:
            9. u ← G.adjacentVertex(t, e)
            10. if u has not been visited, then
                11. u.level = t.level + 1
                12. u.score = t.score * u.rank
                13. update the label of u.
                14. enqueue u onto Q
   15. 

Procedure 1 first removes sink and source nodes from the graph. Afterwards, the BFSLabel procedure (Procedure 2) is called. Similar to the kDLS approach, Procedure 2 broadcasts information in two directions: the outgoing and incoming adjacent vertex lists. The vertex level is used to preserve distance as the information is broadcast as well as determine when k is reached. However, unlike kDLS, the BFSLabel procedure will
multiply vertex semantic ranks into scores and include them in the vertex labels. As a result, Procedures 1 and 2 build vertex labels with information from their semantic networks. Once the broadcast is completed for a vertex, the labels for all those involved are updated accordingly with the vertex subsequently being removed from the graph.

2.2 Indexing Time Complexity Analysis

We let \( V \) represent vertices and \( E \) represent edges in a graph \( G \). Procedure 1 begins by iterating over all vertices in the graph to first remove sink and source vertices, which is simply \( O(V) \). From line 8, for each vertex the BFSLabel procedure is called. Since BFSLabel itself follows a standard BFS procedure, and the label update will involve no more than \( |V| \) labels, the worst case the index construction time is \( O(|V|(|V|+|E|)+|V|^2\log|V|) \). The term \( |V|(|V|+|E|) \) is derived from the BFS procedure, while the term \( |V|^2\log|V| \) is a result of the label update and includes post-sorting the label indexes for efficient use in the future.

However, this worst case analysis does not take into consideration two important factors that significantly reduce the actual running time. (1) BFS is bounded by a distance \( k \). In practice, this restriction reduces the average time it would take to run on a graph the size and density of the UMLS considerably. (2) The label size of each vertex is much smaller than \( |V| \), especially for vertices that are removed earlier.
Chapter 3: System Framework for Online Queries Using Indexing

The cost to load the kDLS index is a major limitation for kDLS to be widely used in the biomedical domain. Typically, it requires more than 20GB memory and takes several hours to load the kDLS index into memory before it can be used to answer queries and generate results. To provide a practical solution for knowledge discovery on the UMLS, an online knowledge discovery platform using Graph indexing was developed. We refer to it as “UMLS-onGrid” or simply “onGrid” if the context is clear. onGrid provides a very user-friendly web interface to query and discover knowledge from the UMLS, and is expected to support future graph indexing engines on the UMLS. onGrid is currently hosted on a CentOS 5.9 release Linux server with approximately 192GB memory and twenty-four AMD Opteron processors.

3.1 The Client-Server Framework

The general framework of onGrid consists of two major components. The first handles interaction with the user, implemented in JavaScript and PHP, while the second handles the processing of user-submitted queries, implemented in C++. The first component communicates requests from the user to the server, in which the second component then executes the request and sends the results back to the first. This design pushes the light and fast pre and post-computation tasks to the client, which has limited resources, and the computing intensive tasks to the server.
Due to the large amount of time required for the server-side program to load the graph index into memory, it would not be feasible to simply execute a new process when a request is made. The program must already have the index loaded in memory, waiting for new requests. Thus, a MySQL database is used to facilitate the communication between the two major components. All requests and results are posted to the database which is regularly checked by both the server and client-side programs. The flowchart of the system framework is illustrated in Figure 2.

![Flowchart of the onGrid Framework](image)

Figure 2. Flowchart of the onGrid Framework

### 3.2 Handling Concurrent Requests

Because the server-side program is a single process, a thread dispatch mechanism is implemented to handle concurrent requests. When a request is made, a thread is created and assigned to a single processor on the server. To provide an upper limit on the number of concurrent requests, a table in the database maintains a record of which processors are
in use, as well as which tasks and threads occupy them. Upon completing the request, the thread terminates and frees its corresponding record. Thus, before communicating with the server-side program, the client can determine if there is an open slot for the request to occupy. If all records are occupied when a new request is made, the request is declined and the user receives a message conveying a heavy server load.

A benefit of the thread dispatch mechanism is that there is no danger of deadlock. Each thread operates independently, and if failure occurs, the system can continue normally. However, it is possible that the thread fails after occupying a slot and before releasing it, and will thus make it inaccessible for future requests. In this case, it would be straightforward to manually check the database record, determine if a request has failed, and reset the record.

3.3 Query Types

There are two primary types of requests. However, all queries require the input of some biomedical concept or its corresponding CUI. To enable natural language based queries, LDPMAP [26], a layered dynamic programming approach that maps a biomedical concept to a UMLS concept, was integrated into onGrid. LDPMAP has been shown to be more effective than the UMLS Metathesaurus and MetaMap in mapping biomedical concepts to a UMLS concept. In this work, biomedical terms in queries are first mapped to UMLS concepts before the query is executed, allowing for more user-friendly queries to be made.
Path Queries and Visualization

Querying for relationships between two concepts returns a collection of paths between them. To provide users an intuition regarding the path results, onGrid utilizes the D3 JavaScript library to visualize the shortest among these paths. Visualizing all paths may not be feasible because results often contain thousands or more paths, which would be hardly discernible given the visual clutter. To visualize the shortest paths between two vertices \( u \) and \( v \), all vertices that have the same distance to vertex \( u \) (or \( v \)) are organized into a set \( S_k \) where \( S_k = \bigcup_{p \in P_{S}(u,v)} \{ x | x \in p, distance(x, u) = k \} \) \( (P_{S}(u,v) \) is the set of shortest paths among the collection of paths between \( u \) and \( v \)). All vertices in a set \( S_k \) will be visualized on a line perpendicular to the line connecting \( u \) and \( v \). In this way we are not only able to observe paths connecting two vertices, but also the shared vertices and edges among those paths.

Figure 3 provides an illustration of such a network of structured paths between Pamidronate Disodium (a drug) and Breast Carcinoma. Users can observe an edge’s semantic type by hovering their mouse over an edge (e.g., Pamidronate Disodium – “may treat” – Paget Disease Extrammamary), or by simply selecting the option to display all edges.
Figure 3. An illustration of visualized paths between two UMLS concepts

In the current version of onGrid, the settings include neighborhood search range, sink and source vertex handling and semantic restrictions, following the preliminary study described in [2] which demonstrates that this setting is cost-effective in UMLS knowledge discovery. In this setting, since k is configured as 6, the system guarantees to find exact distances no more than 6 hops, or at least one shortest path no more than 6 hops, in the underlying graph.
Large Scale Relationship Matrix Generation

To measure the closeness between two concepts, onGrid takes into account the semantic type of each concept (vertex). Abstract semantic types are more likely to be related to a large amount of concepts and therefore are considered weak relationships. To emphasize the more concrete concepts in a path, the closeness between two concepts is measured by:

\[ R(u,v) = \sum_{p \in P(u,v)} \prod_{x \in p} g(x) \]

Where \( P(u,v) \) is the collection of all paths between \( u \) and \( v \) discovered by kDLS, excluding paths with lengths equal to 1. \( g(x) \) is the semantic function on vertex \( x \). In the onGrid implementation, \( g(x) = 1/h \) where \( h \) is the reverse topological level of the semantic type associated with vertex \( x \). Under this measurement, two concepts are likely to be close if there are many short and concrete paths between them.

Using this concept similarity measurement, users can input two sets of biomedical concepts (or CUIs) to view both a relationship and distance heatmap/matrix (as illustrated in Figure 4).
Figure 4. An example of a relationship and distance matrix generated by onGrid

The distance heatmap provides a distance between every two concepts. However, distance alone may not be a good measurement for the relationship between concepts. Thus, onGrid provides the relationship heatmap using the concept relationship measurement function $R(u, v)$ defined above, which extends the measurement in [2] by giving more weight to concrete paths, i.e., paths with less abstract concepts. Similar to [2], paths with only one edge (i.e., direct relations) are not counted in $R(u, v)$ to avoid
bias towards existing knowledge. Finally, onGrid provides a convenient feature for exploring these two matrices: If a user is interested in any particular CUI pair, he/she may click the corresponding unit and onGrid will display the result of the shortest path query on them.

onGrid also supports knowledge discovery on large sets of UMLS concepts for users who wish to utilize this functionality due to the large amount of processing time required. OnGrid supports these types of applications by allowing users to upload two files such that each represents a set of concepts, track their jobs and download the results (a valid email address is required for notification purposes).
Chapter 4: onGrid Knowledge Discovery

A primary feature of onGrid is to study the relations in a set of biomedical concepts, or between two sets of biomedical concepts. In the following two case studies, we demonstrate the effectiveness of this feature by generating relationship matrices for particular datasets of interest, use cross-validation, and analyze their use towards clinical applications.

4.1 Disease-Gene

In the first case study, the focus was on disease relationships in the OMIM (Online Mendelian Inheritance in Man) ontology dataset. OMIM is a database collection of knowledge diseases with a genetic component. onGrid was used to generate a relationship matrix between diseases in OMIM and genes in HUGO (Human Genome Organization). Then, given a threshold $\delta$, the relationship matrix was converted into a 0-1 relationship matrix. Weighted relations $S$ were constructed over OMIM diseases by the number of genes shared by disease pairs in the 0-1 relationship matrix. To cross-validate the results, the same weighted disease relations $T$ were built on CTD, the comparative toxicogenomics database. Folder enrichment was used to measure the results. The folder enrichment function is defined as

$$\alpha = \frac{|S'(\alpha)|}{|T(\alpha)|}$$

where $S' = S \cap T; S'(\alpha)$ is the number of elements in $S$ that are ranked in the top $\alpha$ percent of $T$ according to the weight of
disease pairs; $T(\alpha)$ is the number of elements in $T$ that are ranked in the top $\alpha$ percent of $T$. It is quite intuitive that $f(\alpha)$ will be around 1 if $S$ is random, and if $f(\alpha)$ is much larger than 1, it suggests that $S$ is statistically significant with respect to $T$.

Figure 5. Max Folder Enrichment $f(\alpha)$ at thresholds $\delta$ ranging from 0.45 to 1.4

Figure 6. Folder Enrichment $f(\alpha)$ at thresholds $\delta$ for both kDLS and onGrid
<table>
<thead>
<tr>
<th>Threshold Level</th>
<th>kDLS δ</th>
<th>onGrid δ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.73</td>
<td>0.45</td>
</tr>
<tr>
<td>2</td>
<td>0.80</td>
<td>0.50</td>
</tr>
<tr>
<td>3</td>
<td>0.87</td>
<td>0.55</td>
</tr>
<tr>
<td>4</td>
<td>0.94</td>
<td>0.60</td>
</tr>
<tr>
<td>5</td>
<td>1.01</td>
<td>0.65</td>
</tr>
<tr>
<td>6</td>
<td>1.08</td>
<td>0.70</td>
</tr>
<tr>
<td>7</td>
<td>1.15</td>
<td>0.75</td>
</tr>
<tr>
<td>8</td>
<td>1.22</td>
<td>0.80</td>
</tr>
<tr>
<td>9</td>
<td>1.29</td>
<td>0.85</td>
</tr>
<tr>
<td>10</td>
<td>1.36</td>
<td>0.90</td>
</tr>
<tr>
<td>11</td>
<td>1.43</td>
<td>0.95</td>
</tr>
<tr>
<td>12</td>
<td>1.50</td>
<td>1.00</td>
</tr>
<tr>
<td>13</td>
<td>1.57</td>
<td>1.05</td>
</tr>
<tr>
<td>14</td>
<td>1.64</td>
<td>1.10</td>
</tr>
<tr>
<td>15</td>
<td>1.71</td>
<td>1.15</td>
</tr>
<tr>
<td>16</td>
<td>1.78</td>
<td>1.20</td>
</tr>
<tr>
<td>17</td>
<td>1.85</td>
<td>1.25</td>
</tr>
<tr>
<td>18</td>
<td>1.92</td>
<td>1.30</td>
</tr>
<tr>
<td>19</td>
<td>1.99</td>
<td>1.35</td>
</tr>
<tr>
<td>20</td>
<td>2.06</td>
<td>1.40</td>
</tr>
</tbody>
</table>

Table 1. Corresponding thresholds δ for kDLS and onGrid

From Figures 5 and 6 we can see that folder enrichment values are much larger than 1. They generally increase when the threshold δ increases. This is because when the threshold δ is high, only the disease pairs which share most of the genes are left for study. However, the thresholds in this study were set to an upper limit to avoid studying too few disease pairs. Notice also that these values become smaller as the percentage α increases. This is understandable because according to the definition, when α increases, the difference between the numerator and denominator tends to get smaller, and f(α) = 1 when α = 100. These folder enrichments tests suggest that the disease pair results
obtained by onGrid are statistically significant in the cross validation with the external dataset CTD.

In addition, Figure 6 includes corresponding results generated from the original kDLS algorithm (indicated by dashed lines). To ensure the results are comparable, the percentiles for minimum and maximum thresholds δ in the onGrid relationship matrices were obtained and used to determine appropriate minimum and maximum δ values for the kDLS matrices. Table 1 lists their respective δ value for each threshold level. We can see that onGrid tends to generate higher folder enrichment values for each respective α, suggesting that incorporating semantic types leads to more focused and correlated diseases and genes.

It is possible to further study a particular disease of interest by observing its most related diseases according to shared genes. As examples, consider Adenocarcinoma of lung and Glioblastoma. The top ranked diseases related to these two diseases are presented as circular arc graphs in Figures 7-10.
Figure 7. Top diseases related to Adenocarcinoma of lung by on Grid ($\delta=1.1$)
Figure 8. Top diseases related to Adenocarcinoma of lung according to CTD
Figure 9. Top diseases related to Glioblastoma by onGrid ($\delta = 1.1$)
Figure 10. Top diseases related to Glioblastoma according to CTD

From Figures 7-10 we can see that the disease relationship generated by onGrid has greater weight variation (visualized by edge thickness) in comparison with disease relationships observed in CTD. In addition, the top related diseases by onGrid are primarily leukemia and carcinoma for Adenocarcinoma of lung, and mostly carcinoma for Glioblastoma. These are consistent with the nature of the two diseases. However, the
top diseases related to CTD are quite general. This suggests that the knowledge discovery results from onGrid are better suited for studying such disease relationships.

4.2 Gene-Gene

For the second case study, the whole gene list extracted from the HUGO data source in the UMLS, which contains 31205 genes, was used as input to onGrid to build the relationship matrix among these genes. This dataset was selected because gene-to-gene relationships have been well studied and can be cross-validated easily.

<table>
<thead>
<tr>
<th>Cluster size rank</th>
<th>Number of genes</th>
<th>Molecular Function</th>
<th>Biological Process</th>
<th>Cellular Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>429</td>
<td>sequence-specific DNA binding transcription factor activity (9.049E-186)</td>
<td>transcription from RNA polymerase II promoter (8.434E-165)</td>
<td>transcription factor complex (1.097E-103)</td>
</tr>
<tr>
<td>2</td>
<td>370</td>
<td>receptor binding (2.247E-70)</td>
<td>regulation of cell proliferation (2.938E-82)</td>
<td>extracellular space (6.136E-53)</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>protein serine/threonine kinase activity (2.939E-72)</td>
<td>protein phosphorylation (1.164E-36)</td>
<td>microtubule organizing center (7.012E-6)</td>
</tr>
<tr>
<td>4</td>
<td>33</td>
<td>transporter activity (5.739E-18)</td>
<td>transmembrane transport (2.321E-6)</td>
<td>hemoglobin complex (2.973E-15)</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>chemokine activity (8.145E-77)</td>
<td>taxis (2.577E-38)</td>
<td>extracellular space (2.717E-37)</td>
</tr>
</tbody>
</table>

Table 2. Top 5 gene clusters and their enrichments using ToppGene
After generating the relationship matrix, gene clusters were extracted by (1) cutting off edges with low weights and (2) applying the graph sparsification algorithm [27] to remove edges that are unlikely to be within the same cluster.

Table 2 shows the enrichments of the top five largest gene clusters (among 118 clusters containing at least two genes) we obtained by applying the above clustering procedure with a weight cut off at 0.6 and 10% global sparsification. One interesting observation is that they are highly enriched with molecular functions, biological processes, and cellular components. This suggests that the gene clusters were divided into highly correlated components. If we look into these gene clusters, we will have many opportunities to find new hypotheses. For example, “SRC, ARHGEF5, KIT, KRAS, ROS1, MET, ERBB2, PARD6A” is a gene cluster obtained above. Through enrichment tests, we found that these are enriched under “intracellular protein kinase cascade” or “enzyme linked receptor protein signaling pathway” except for the ARHGEF5 gene. By applying kDLS path queries between the ARHGEF5 gene and the other genes, we found that there are several paths available between them. Many of these paths include signaling protein and malignant neoplasm of breast which can serve as hypotheses to be verified in the future.

From searching literature, a study was found which identifies ARHGEF5 as a SRC SH3-binding protein [28], which partially reveals the relationship between ARHGEF5 and SRC.
Chapter 5: Conclusion

UMLS-onGrid provides an efficient web-based platform to perform graph queries and knowledge discovery on the UMLS. The preliminary work on kDLS was extended to incorporate the UMLS semantic network to refine the knowledge discovery process, and is used as the server side index engine for onGrid. As demonstrated, the indexing construction time complexity is reasonable for a graph the size and density of the UMLS, so that it is easy to update future versions of onGrid.

The primary features of onGrid are efficiently generating shortest paths between two UMLS concepts to be visualized as well as building relationship and distance heatmaps. The relationship heatmap enables researchers to quickly identify highly related biomedical concepts and directly check the transitive relations between any two concepts by clicking the corresponding unit. By using the upload feature to generate relationship and distance matrices for large datasets, users can perform data mining to find highly connected components. The knowledge discovery results in both case studies demonstrate the effectiveness of using onGrid in studying biomedical concept relations.

In particular, results generated from onGrid were shown to be improvements over those generated by the original kDLS in cross-validation with an external dataset. onGrid is expected to be used for many applications to assist knowledge discovery in the biomedical field.
Bibliography


2. Xiang, Y., Lu, K., James, S., Borlawsky, T., Huang, K., Payne, P.: k-neighborhood Decentralization: A Comprehensive Solution to Index the UMLS for Large Scale Knowledge Discovery. Journal of Biomedical Informatics (in press)


7. Hristovski, D., Rindflesch, T., Peterlin, B.: Using literature-based discovery to identify novel therapeutic approaches. Cardiovascular \\& Hematological Agents in Medicinal Chemistry (Formerly Current Medicinal Chemistry-Cardiovascular \\&


the Efficacy of Natural Language Processing Techniques for Validating Translational Knowledge Constructs. In : Summit on Translational Bioinformatics, p.95 (2009)


