An Algorithm for Combining Graphs Based on Shared Knowledge

Hisham Al-Mubaid and Said Bettayeb
University of Houston- Clear Lake, Houston, TX USA
Hisham@uhcl.edu & Bettayeb@uhcl.edu

ABSTRACT.
We propose an algorithm for connecting nodes from multiple disconnected graphs based on a given tuple set representing shared knowledge. The set of tuples is used to create bridge-edges for combining two graphs. The path from a node in a graph to a node in the other graph passes through a bridge-edge. This method of combining two graphs will enable more comprehensive understanding and exploring of the relatedness of the knowledge entities (the nodes) in two graphs based on a given domain knowledge represented in the set of tuples. This approach has useful applications in various domains and in particular in bioinformatics. In bioinformatics, for example, we can explore the functional relationship between two gene products given their Gene Ontology annotation terms from the molecular function MF and biological process BP graphs of GO. Moreover, the proposed algorithm can be applied to WordNet to enable exploring the relative degree of relatedness of words from multiple lexical hierarchies, like nouns and verbs, within the WordNet.

Keywords
Combining graphs, graph theory, gene ontology.

1. Introduction

There are various ways to represent structured knowledge and information in different domains. One of the common ways to represent structured knowledge is using graphs [1,6]. Typically, ontologies and taxonomies are represented as trees or hierarchical graphs similar to trees in which the knowledge is embedded in the nodes and edges. The edges denote the relationships between the information constructs which are the nodes. For example, *WordNet* is the main source of structured knowledge about nouns, verbs, adjectives and adverbs in the English language and can be viewed as hierarchical graph [13]. WordNet is a lexical database of English terms organized, in its simplest way, into four graphs for *nouns*, *verbs*, *adjectives* and *adverbs*. A large number of algorithms and techniques have been devised specifically for WordNet to examine and estimate the degrees of similarity and relatedness between terms for various applications including word similarity, information retrieval, and knowledge discovery [5, 13]. Also, the bioinformatics field is highly loaded with knowledge sources that are structured into graphs and trees, e.g. GO, MeSH, SNOMED-CT, HPO [1, 6, 7, 10, 11, 12]. In some applications, it is important to connect nodes from multiple graphs, perhaps within the same ontology, to produce the desired results.

In this paper, we propose an algorithm for connecting nodes from multiple disconnected graphs, with disjoint vertex sets, based on a given tuple set that summarizes domain knowledge from the same domain of the graphs. Each graph represents certain aspect of knowledge structured in a way that the edges represent the relationships between the knowledge terms or entities (*i.e.*, the nodes).

Since the two graphs represents two aspects (*e.g. nouns* and *verbs*) of the knowledge source (*e.g. WrodNet*) then we can utilize a shared knowledge (*e.g.* benchmark corpus), represented as *tuples*, to connect the two graphs in a systematic and structured way. The set of tuples is used to create new edges connecting nodes from the two graphs. Such edges are called "*bridge-edges*". Therefore, the path between two nodes in two graphs can pass through one or more *bridge-edges*.

This method of connecting two graphs, or two knowledge sources, will enable us to explore and understand the degree of relatedness of the nodes in two graphs more comprehensively based on a given domain knowledge summarized in the set of tuples.

This approach has many useful applications in various domains and in particular in bioinformatics. In bioinformatics, there exist massive volumes of data and information from medicine and molecular biology organized and structured into a large number of ontologies, taxonomies, vocabularies, and other hierarchical structured databases [16]. The most commonly used biomedical ontologies are represented as directed acyclic graphs (DAGs) [8, 9, 11]. For example, we can explore the functional relationship between two genes given their gene ontology (GO) annotation terms (GOA terms) from the *molecular function* (MF) and *biological process* (BP) taxonomies combined [7, 12]. Moreover, the proposed algorithm can be applied to *WordNet* lexical database to enable

exploring the relatedness of words from multiple lexical hierarchies, e.g. nouns and verbs, within the WordNet [5, 13]. Furthermore, another application of this method of connecting two knowledge graphs is ontology integration [10]. Ontology integration has been an important research topic in the past few decades, and it is used for information retrieval and knowledge extraction from multiple ontologies sources in many domains [10].

The focus of this paper is on graphs that represent structured knowledge such as ontology, terminology, or simply a structure knowledge graph. The method for connecting two graphs based on shared domain knowledge is detailed in Section 3 and explained in an algorithm in section 3.2.

2. Related Work

Graphs are abstraction of real life problems. Graphs have long been used in many applications to many fields in computer science, most branches of engineering, chemistry, telecommunications, scheduling, transportation, and social systems to name just a few [2, 3, 4, 8, 10]. A graph G = (V, E) is a pair where V is a set of objects called *nodes*, or *vertices*, and E a set of *edges*. Graphs are generally represented by means of diagrams in which vertices are represented by small circles or dots, and edges by line segments. If the graph is directed then the edges are represented by arrows. The edges of a graph can have weights that could represent the distance between the nodes, the time that it takes to traverse that link, the probability that that link does not fail, or any other measure relevant to the problem at hand. If the edges have weights, the graph is said to be a weighted graph and is represented by a triplet G=(V,E,W) where W is the weight on the edges.

Many ways of combining graphs to produce new graphs have been proposed in the literature. The union, the cartesian product, and the join are just few examples of graph combinations. Given two graphs $G_1=(V_1, E_1)$ and $G_2 =(V_2, E_2)$ where V_1 and V_2 are distinct, the cartesian product $G=G_1xG_2$ is the graph whose set of vertices is the Cartesian product of V_1 and V_2, *i.e.* $V=\{(u_1,u_2)$ such that $u_1 \in V_1$ and $u_2 \in V_2\}$ and the set of edges E consists of the edges $((u_1, u_2)$ $(v_1, v_2))$ such that $u_1=v_1$ and $(u_2 ,v_2) \in E_2$ or, $u_2=v_2$ and $(u_1, v_1) \in E_1$. The join $G= G_1+ G_2$ is the graph whose set of vertices V is the union of V_1 and V_2 and the set of edges E is the union of E_1, E_2 and the set $\{uv \mid u \in V_1$ and $v \in V_2\}$; i.e. besides the edges that are already in G_1 and G_2, an edge is added between every node in G_1 and every node in G_2 [2].

In this paper, we propose a way of combining graphs that is similar in nature to the join operation [15]. The set of vertices is obtained in exactly the same fashion, *i.e.* $V = V_1 \cup V_2$. The set of edges E differ from that of the join in the last component only, *i.e.* $E= E_1 \cup E_2 \cup E_C$ where E_C is defined as follows. There is an edge between vertex x of G_1 and vertex y of G_2 if x and y appear in the same tuple of the domain knowledge. The weight of that edge depends on the number of tuples that contain such a pair. We call such an edge a *bridge-edge*. All the other edges are assigned weight 1. Shortest path problem is the one of the most primary graph traversal problem and arises in a wide variety of applications. Transportation and telecommunications, are just a few examples of such applications [3, 4]. Finding the most reliable path for a communications is reduced to finding the shortest path. Bellman-Ford and Dijkstra's algorithms are two efficient algorithms that compute single-source shortest paths in a weighted graph [3]. The distance-vector routing protocol uses a variant of Bellman Ford algorithm.

3. Connecting Graphs

We would like to connect two graphs representing two aspects of structured knowledge source in a given domain. In the general form, a graph consists of nodes where each node is a *term* or *entity* in a domain knowledge, and the edges depict the relationships, *e.g. is_a*, between the nodes. We assume that the graph is a directed acyclic graph (DAG). Typically, DAGs are used as a common way of representing knowledge sources like ontologies. A DAG is a graph that has no cycles and each edge has a direction. A knowledge source usually relies on viewing the DAG as a hierarchical graph with *root* node, *internal* nodes, and *leaf* nodes. As we go down the graph from the root towards the leaves, the nodes, or *terms*, become more specific and the root is the most general knowledge term. Two graphs will be connected using domain knowledge presented as a set of tuples TP ={P1, .., Pn}

3.1 Problem formulation

Given two DAG graphs G1 = (V1, E1) and G2 = (V2, E2). As stated earlier, we assume that the set of vertices V_1 and V_2 are distinct, G1 is connected and so is G2, and no edges exist between G1 and G2. Each edge eij has weight 1: W(eij)=1 s.t. eij is an edge in G1 or in G2.

Also given a set TP of *n* tuples s.t.

$$TP = \{P1, \ldots, Pn\}.$$

Each tuple Pi (i=1, .., n) is a set of terms

$$Pi=\{t_1,\ldots,t_k\}$$

where each term t_j is a term (*a node*) in G1 or in G2. See Figure 1. Further, each tuple Pi includes terms (nodes) from both graphs G1 and G2. That is,

$$Pi=\{\ldots,t_p,\ldots.,t_q,\ldots\}$$

where t_p is a node in G1 whereas t_q is a node in G2, or vice versa.

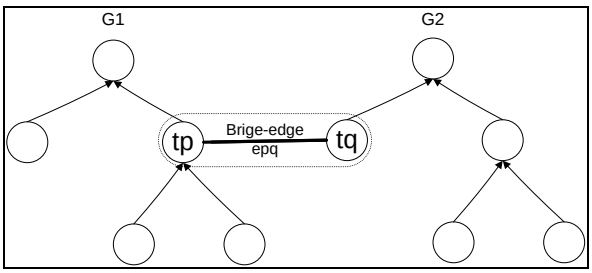

Figure 1: A *bridge-edge* and bridge nodes

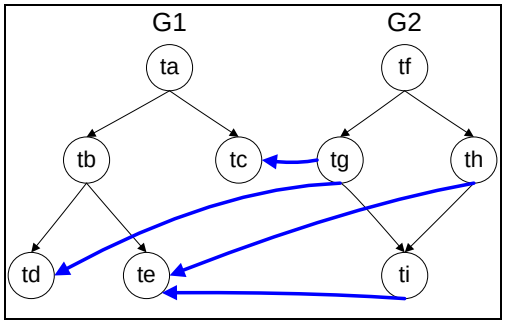

Figure 2: Four bridge edges between G1 and G2

For each pair of terms (t_p, t_q) such that t_p and $t_q \in$ Pi (Pi is some tuple in TP), and t_p,t_q do not belong to the same graph: We draw an edge e_{pq} to connect node t_p and t_q (i.e., connecting graph G1 with graph G2). The edge e_{pq} is called *bridge-edge* because it has one endpoint in G1 and the other in G2, and the nodes t_p and t_q are called b_{eq}. That is, the node t_p in G1 is also called b_{pq} because it is part of bridge $\mathbf{e_{pq}}$ to node t_q in graph G2.

Define a *b_count(t_p, t_q)* to be the number of tuples in TP containing both terms t_p and t_q where t_p, t_q are not in the same graph:

b_count(t_p, t_q) = number of tuples containing both terms t_p and t_q……..….(1)

We also define a weight function *w*(.) for each newly introduced bridge edge as follows;

$w(e_{pq})$ = b_count(t_p, t_q)/max(b_count())……..(2)

Thus, from the set TP of n tuples, we create a set of bridge edges b_{kl} between G1 and G2 and each such bridge-edge b_{kl} has a weight $w(b_{kl})$ such that, according to equation (2):

$0 < w(b_{kl}) \leq 1$……..(3)

P1 = {t_c, t_g}
P2 = {t_c, t_d, t_g}
P3 = {t_e, t_h, t_i}
P4 = {t_a, t_c, t_e}
P5 = {t_e, t_h}

Figure 3: The tuple set for the example graphs in Figure 2.

These bridge edges can be represented in a table such that rows are the nodes in G1 and the columns are the nodes in G2 as in Table 1.

For example, as shown in Table 1, the bridge edge between node t_p in G1 and node t_q in G2 is 0.7 while the weight of the bridge edge between tp+1 and tq+1 is 0.4. Figure 2 illustrates four bridge edges between two graphs G1 and G2, namely (t_c, t_g), (t_d, t_g), (t_e, t_h), and (t_e, t_i). The four bridge edges in Figure 2 are created from the tuples shown in Figure 3.

3.2 An algorithm for combining graphs

The following algorithm combines two graphs G1 and G2 according to a given set of tuples.

Algorithm: Combine_two_graphs

Input: 1) Two DAG graphs G1=(V1, E1) and G2 = (E2, V2). V1 = {$t_i \mid 1 \leq i \leq n1$}, V2 = {$t_j \mid 1 \leq j \leq n2$ }

2) A set TP of *n* tuples {T1, .., Tn} s.t. each tuple Ti is a set of terms: Ti = {t_1, .., t_k}

3) A threshold *tsh: 0 < tsh ≤ 1*

Output: A graph Gc=(Vc, Ec): Gc is a graph resulting from connecting nodes in G1 with nodes in G2

1. Initialize *b_count(tp, tq)* = 0 for all $1 \leq p \leq n1, 1 \leq q \leq n2$; also initialize Ec = ∅
2. For each tuple Ti={t_1, …, t_k} do the following:
   2.1 For each pair {(t_p, t_q) | t_p,$t_q \in$Ti and $t_p \in$V1 and $t_q \in$V2} set
   b_count(t_p, t_q) = b_count(t_p, t_q) +1
3. Compute *max_bc = max{b_count(t_p,t_q) } for all t_p, t_q*
4. For each pair (t_p, t_q) s.t. *b_count(tp, tq)* > 0, compute
   w(e_{pq}) = b_count(t_p,t_q)/max_bc
5. For each pair (t_p, t_q), if w(t_p, t_q) ≥ tsh
   then Ec = Ec U e_{pq} (note: e_{pq} is an edge between t_p and t_q)
   else w(t_p, t_q) = 0
6. Output combined graph Gc = (Vc, Ec) such that
   Vc = V1 ∪ V2 and Ec = E1 ∪ E2 ∪ Ec

3.3 Graph exploration and terms relationship

A common technique to explore a graph representing knowledge source, like ontology or vocabulary, is to examine the relationships between the nodes in the graph. The basic way to estimate the relationships

between the terms of the knowledge is though path length as a measure of relationship. Path length as a relationship measure between the terms in a given ontology graph has been used extensively in *WordNet* and in bioinformatics domains [5, 6, 13, 9, 13, 14]. The path length between two terms in the same graph is computed straight forward by edge counting. If there is more than one path, then the shortest path is taken as follows:

$$PL(t_1, t_2) = \text{the shortest path length (least number of edges) between nodes } t_1 \text{ and } t_2, \quad \ldots\ldots(4)$$

where t_1 and t_2 are two nodes in a single graph (G1 or G2). If the two nodes belong to two graphs, then each path between them passes through a bridge edge. Define a path length between two nodes belonging two graphs as follows: If t_i and t_j are not in the same graph then:

$$PL(t_i, t_j) = PL(t_i, t_p) + PL(t_j, t_q) + 1/w(e_{pq}) \quad \ldots\ldots(5)$$

where t_p is a bridge node in the path from t_i to the root, and similarly t_q is a node in the path from t_j to the root; and w(.) is weight function shown in equation (2). This way we can estimate the relationship between concept or term t_i and term t_j belonging to two different graphs.

4. Applications and Case Studies

4.1 Case study: WordNet

WordNet is the most comprehensive lexical database of the English language. WordNet consists of four independent tree-like hierarchies for *nouns*, *verbs*, *adjectives* and *adverbs* [13]. The nodes represent concepts, or set of terms called *synsets* and the edges represent lexical and semantic relationships between them. In the two graphs of *WordNet* shown in Figure 4. If we are given a tuple set of English words, we can connect the two graphs via *bridges*. Then we can estimate and compute the similarity and relationships between a noun (*e.g. car*) and a verb (*e.g. drive*). We can use a standard benchmark corpus of words (e.g. *Brown* corpus of *WSJ* corpus) and divide it into chunks of 5 words or 10 words (i.e., window size *w=5* or *w=10* respectively) and use them as tuples to connect the two graphs. The Brown corpus is a one million words benchmark text corpus and has been extensively used in natural language processing research and it can be used here to produce tuple sets to connect graphs. For instance, from the tuple {they, can, build, beautiful, house} there will be a bridge to be created between the noun node {*house*} in the noun sub-graph and the verb node {*build*}in the verb sub-graph.

	….	t_q	t_{q+1}	….
…				
t_p		0.7		
t_{p+1}			0.4	
…				

Table 1: weights of bridge edges

4.2 Case study in bioinformatics

The second application example we discuss is in the bioinformatics domain, specifically using the gene ontology (GO) [7]. We can explore the functional relationship between two genes given their gene ontology annotation terms (GOA terms) from the *molecular function* and *biological process* graphs combined [1, 6, 12]. Gene ontology is a structured vocabulary of gene functions and related information at the molecular level, biological process and cellular localization. GO, therefore, is composed of three orthogonal sub-ontologies: *molecular function* (MF), *biological process* (BP), and *cellular component* (CC). It is the main source of information about gene functions, processes, and localizations. GO have been studied and investigated for long time, and very extensively, for computing gene similarity and relationships among gene products in various organisms [10, 11, 12]. Moreover, most of the approaches for discovering new gene functions and identifying gene disease associations are also based on GO. However, there has not been any work that explores the functional relationships between gene products in terms of their *MF* and *BP* annotation terms combined. Figure 5 illustrates small parts of the *MF* and *BP* graphs of GO. If we use the gene ontology annotation (GOA) database of one genome as tuple set (shared knowledge), then we can introduce a large number of *bridge edges* between these two graphs. Of course, the GOA database represents a verified domain knowledge and so the bridge edges are valid information augmented into the graphs. The GOA database, for the human genome for example, contains the GO annotation terms for each gene from the *MF*, *BP*, and *CC* components. Table 2 contains the GOA terms of four genes from the MF and BP sub-ontologies. Thus with the GOA database we can connect MF and BP nodes which will enable to compute the functional similarity, using path length, between two genes or two GO terms within the MF and BP space. Currently all the projects and techniques for estimating the functional similarity and relationships between genes, GO terms, gene-diseases, are based on either *MF* or *BP* but not combined [10, 11, 12, 14]. To the best of our knowledge, we have not seen any work that combines MF and BP to compute similarity is such a comprehensive way.

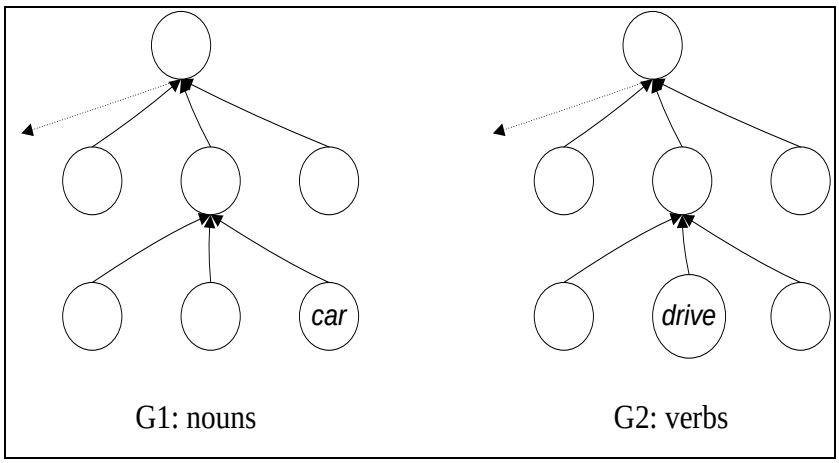

Figure 4: Simplified illustration of two graphs of WordNet: nouns and verbs.

5. Conclusion

We presented a method for combining multiple graphs via *bridge-edges* using shared domain knowledge. The method is useful is cases when each graph represents one aspect of some domain knowledge. Connecting two graphs can enable more comprehensive viewing, exploring, and understanding of the knowledge with its both aspects. The external knowledge is represented as a set of tuples to allow for determining *bridge* nodes and for creating bridge-edges between the graphs. We have discussed two application domains for the proposed algorithm. To the best of our knowledge, this is the first attempt for such a work that enable exploring multiple knowledge graphs by connecting the graphs using meaningful relations derived from domain knowledge.

References

1) J.J. Goeman and U. Mansmann, Multiple Testing on the Directed Acyclic Graph of Gene Ontology. Bioinformatics 2008, 24(4):537-544.
2) G. Chartrand; and L. Lesniak, Graphs and Digraps, 4th Edition, 2004, CRC press.
3) T.H. Cormen, C.E. Leiserson, R. L. Rivest and C.L. Stein, Introduction to Algorithms,3rd Edition, 2009, MIT Press.
4) R. K. Ahuja, T.L. Magnanti, J. B. Orlin. Network Flows. Prentice Hall, 1993.
5) T. Pedersen, S. Patwardhan, and J. Michelizzi, WordNet::Similaritymeasuring the relatedness of concepts, presented at the 19th Nat. Conf. Artif. Intell. (AAAI-04), San Jose, CA, 2004.
6) H. Al-Mubaid and Hoa A. Nguyen. Measuring Semantic Similarity between Concepts within Multiple Ontologies in the Biomedical Domain. IEEE Trans. SMC-C, Vol.39, No.4, pp. 389-398, July 2009.
7) The Gene Ontology: www.geneontology.org
8) R. Ambauen, S. Fischer and Horst Bunke. Graph Edit Distance with Node Splitting and Merging, and Its Application to Diatom Identification. In Graph Based Representations in Pattern Recognition, Lecture Notes in Computer Science, 2003, Volume 2726, 2003.
9) K. M. Sim and P. T. Wong. Toward Agency and Ontology for Web-Based Information Retrieval. IEEE Transactions SMC, vol.34, no.3,2004.
10) Vaida Jakoniene and Patrick Lambrix. Ontology-based integration for bioinformatics. Proc of 31st VLDB Conf, Trondheim, Norway, 2005.
11) Robinson PN, Mundlos S. The Human Phenotype Ontology. Clinical Genetics, 2010: 77: 525–534.
12) H. Al-Mubaid and A. Nagar, A New Path Length Measure Based on GO for Gene Similarity with Evaluation Using SGD Pathways, IEEE CBMS-2008.
13) G. A. Miller, WordNet: A lexical database for English, Commun. ACM, vol. 38, pp. 39–41, 1995.
14) R. Rada, H. Mili, E. Bichnell, and M. Blettner, Development and application of a metric on semantic nets, IEEE Trans. Syst., Man, Cybern., vol. 9, no. 1, pp. 17–30, Jan./Feb. 1989.
15) G. Ganapathy and R. Lourdusamy. Matching and Merging of Ontologies Using Conceptual Graphs. Proc. of the World Congress on Engineering WCE 2011, July, 2011, London, UK.
16) L. Humphreys, D. Lindberg, H. Schoolman, and G. Barnett. The Unified Medical Language System: Informatics Research Collaboration. J. of the Ameri Medical Informatics Association, 1(5), 1998.

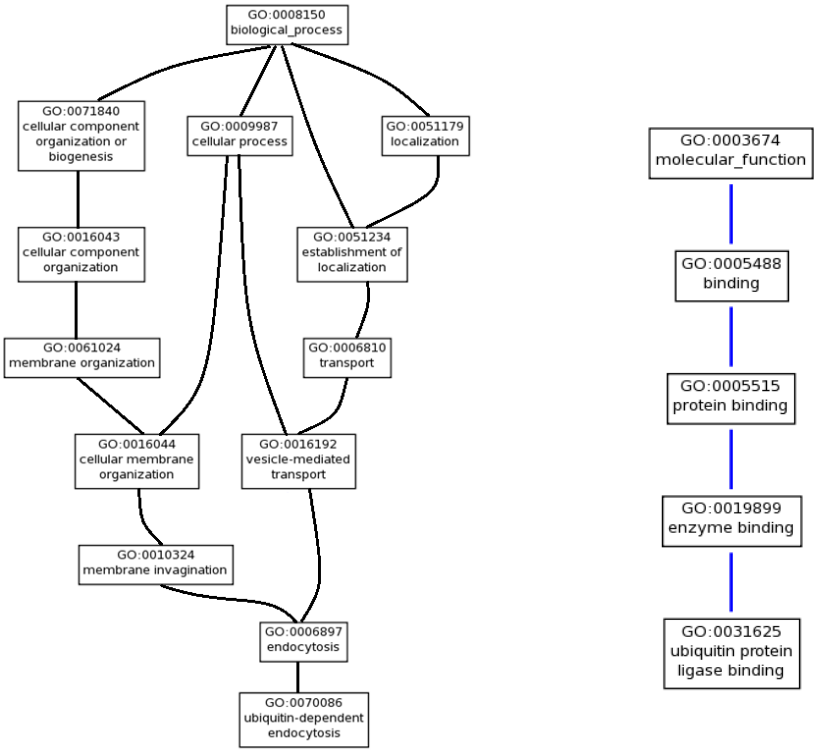

Figure 5: Illustration of parts of the BP and MF aspects of the gene ontology.

Gene id	GO annotation	
	BP terms	**MF terms**
AAC1	GO:0006783, GO:0006810, GO:0006839, GO:0009060 GO:0015886, GO:0055085	GO:0005488, GO:0005215 GO:0005471
AAC3	GO:0006783, GO:0006810 GO:0009061, GO:0015886 GO:0055085	GO:0005488, GO:0005215 GO:0005471
ROD1	GO:0042493, GO:0070086	GO:0031625
SNM1	GO:0006379, GO:0006364	GO:0003723, GO:0000171 GO:0016787, GO:0004518

Table 2: Example of GOA data for four genes.