Finding Lowest Common Ancestor for Given Go Terms

Project Report

Team 2　2021.06.01

龚禹桥 519111910187

姜　戈 519111910306

徐静思 519111910070

Contents

Motivation

Finding LCA is an important step in calculating the semantic similarity between two given genes

Measuring the semantic similarity between Gene Ontology (GO) terms is an essential step in functional bioinformatics research. However, finding lowest common ancestor first is needed in many algorithms of measuring the semantic similarity between GO terms, such as Resnik and his team's algotirhm based on Information Contents, Wu and Palmer's algotirhm based on Hierarchical Structure and etc. Therefore, how to finding lowest common ancestor between given GO terms better is worth reconsidering, which may also helps to improve the efficiency of calculating the semantic similarity.

Solution

1.Multiplication algorithm(倍增算法)

There are four main algorithms for finding the Lowest Common Ancestor(LCA) : general algorithm(朴素算法)、multiplication algorithm(倍增算法)、RMQ algorithm and Tarjan algorithm. Here we choose the multiplication algorithm(倍增算法) for its conciseness and understandability.

2.Linked-forward-star(链式前向星)

Linked-forward-star(链式前向星) is a kind of data structure that can be conveniently used to save plot, which can be perfectly applied to storing GO terms and their relationships as well.

More detailed references:

3.Set every node deepest -- main point of innovation

The hardest part of finding the lowest common ancestor between given Gene Ontology (GO) terms is that the structure of GO terms is definitely not a tree, that is a child node may has more than one parent nodes, resulting in one node with more than one depth. Here in our package, we set every GO term (node) the deepest one as its depth, in order to contain more gene information in the file and turn it into a tree.

4.The pseudocode is as follows.

```
#define N 300
#define M 7
int  n; //save the node number
char *arr[N]; //save the code of each node number
int fa[N][M]; //save the multiplying father of each node
int dep[N]; //the depth of each node
int head[N];
int nx[N];
int to[N]; //this tree array is used to save the GO plot
int tot; //save the edge index
int in[N]; //entering degree of each node

int main(int argc,char *argv[])
{
    initialize all the variables

    for every two lines in file
```

```
        do
        code <- first line
        code has been saved in arr[x]?v=x:v=n,save arr[n]
        for each code in second line
            do
            code has been saved in arr[x]?u=x:u=n,save arr[n]
            to[tot]=v;
            nx[tot]=head[u];
            head[u]=tot++;  //save the plot using chain forward star
            in[v]++;
            done
        done

    if(dep[u]<=d){
        dep[u]=d;
        for(int i=head[u];i;i=nx[i]){
            int v=to[i];
            dfs(v,d+1);  //DFS, make sure all node is the deepest if it has multiple father
        }
    }

   for(i=head[u];i;i=nx[i]){
            int v=to[i];
            if(dep[u]>=dep[fa[v][0]])fa[v][0]=u;
            father0(v);
        } //set the deepest father as its ultimate father

    DFS again
        do
        fa[u][i]=fa[fa[u][i-1]][i-1];
        done

    u=argv[1]
    v=argv[2]

    if(dep[u]<dep[v])swap(u,v) //make sure u is the deeper node
    int d=dep[u]-dep[v];
    for(i=0;(1<<i)<=d;i++){// (1<<i) <=d make sure u don't jump over v
        if((1<<i)&d){   // (1<<i) &d find the suitable span to jump
            u=fa[u][i];
        }
    }
    if(u==v)LCA=u,return;//if u==v, LCA finded
    for(i=M-1;i>=0;i--){
        if(fa[v][i]!=fa[u][i]){
            u=fa[u][i];
            v=fa[v][i];
        }
    }
    LCA=fa[u][0],return
}
```

5.Visualization

To present the relationships between GO terms vividly and help to make our point better, we also write a code in Python that achieves visualization of the data set.

6.More details of our package

All codes and data can be obtained at https://github.com/GYQ-form/bi290-project . Read README.md and the manual for more usage details.

Experiment

As the data of GO terms can be easily obtained from an open and free database http://geneontology.org/ , we determined to test our package on real-world data directly without simulating.

On real-world data

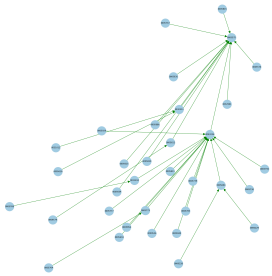
Any GO files (`.obo` format) downloaded from the GO database [http://geneontology.org/] can be analyzed to finding LCA using our method. Here we use the slim GO file [goslim_generic.obo] as a demonstration.

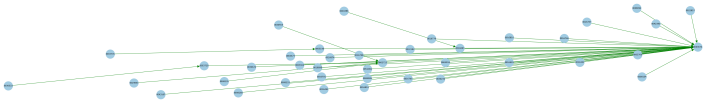
The first step is to extract the chirdren and parents nodes, that is GO IDs in this case, from the file. Type `cat goslim_generic.obo | grep -Po "(?<!_)id: GO:[0-9]{7}|is_a: GO:[0-9]{7}" | awk -f std.awk >slim_extract` on the command line. The bash file is written in std.awk . The extract information is written in slim_extrat.

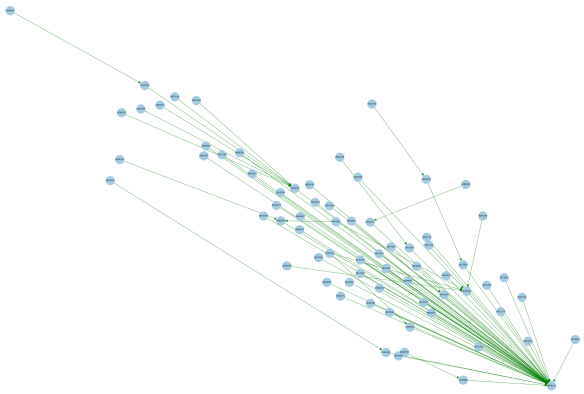
Here are several examples :

```
[u1@localhost project]$ bin/GoJumpLca
usage:<filename> GO_id1 GO_id2
[u1@localhost project]$ bin/GoJumpLca 0000001 0000005
the GO id is not found!
[u1@localhost project]$ bin/GoJumpLca 0000228 0000003
there is no LCA for the two GO terms!
[u1@localhost project]$ bin/GoJumpLca 0008150 0000003
the LCA between two GO terms is:0008150
[u1@localhost project]$ python bin/image_generated.py
please input the name of the file
[u1@localhost project]$ python bin/image_generated.py "slim_extract"
Image generated
```

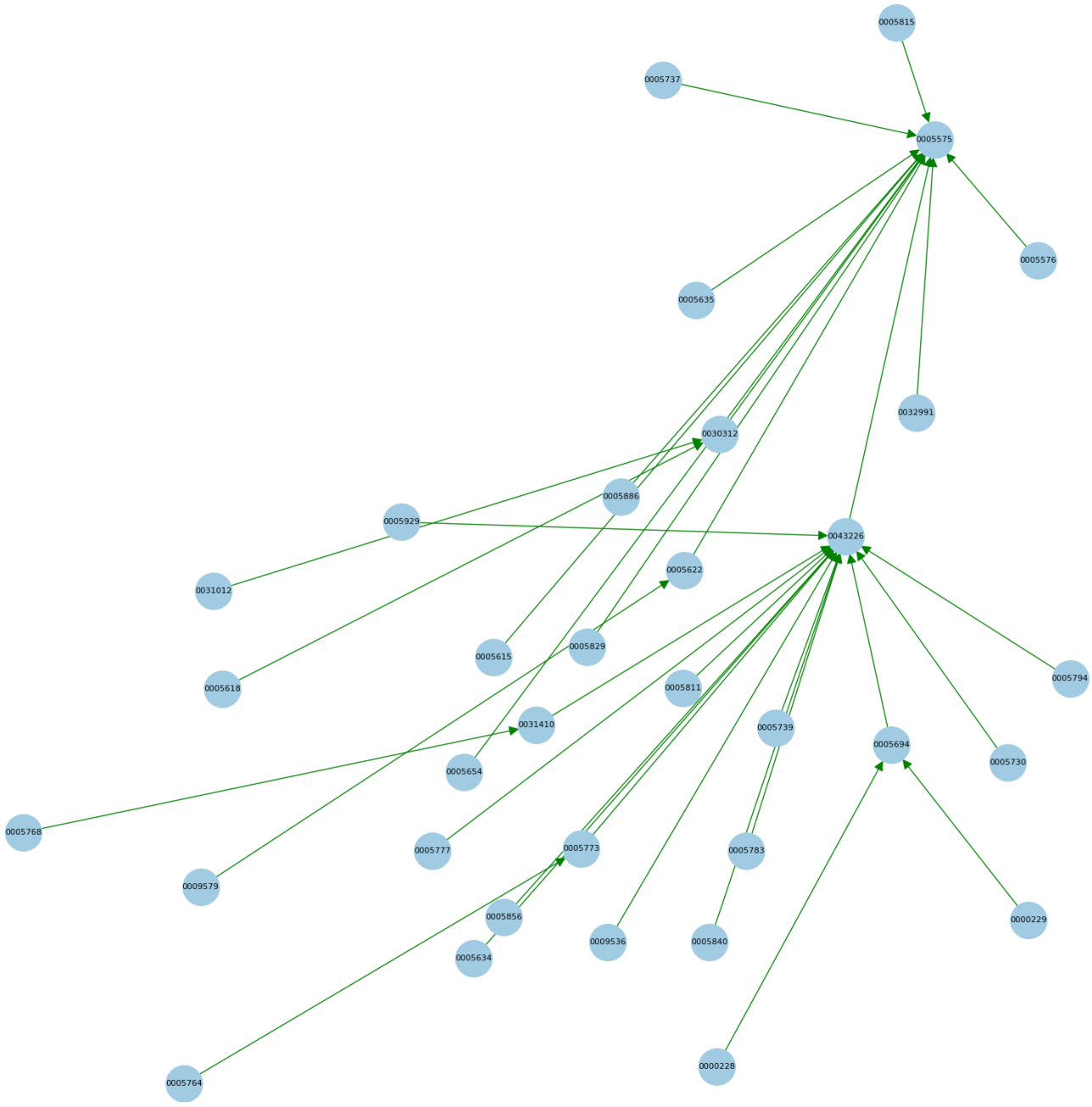
Here is the result of visualization:







Zoom in the image locally:



Supplementary information

To execute our package successfully, type the commands below in order

```
mkdir bin shared lib
cd src
#Add environment variable
export C_INCLUDE_PATH=../include:${C_INCLUDE_PATH}
make
make move
cd ../
#Add environment variable
export LD_LIBRARY_PATH=./lib:${LD_LIBRARY_PATH}
#GO id is arbitrary as long as its length=7
bin/GoJumpLca 0008150 0000003
#generate image
python bin/image_generated.py "slim_extract"
```

Conclusion

Finding Lowest Common Ancestor(**LCA**) is an important step in calculating the semantic similarity between two given genes. There are four main algorithms for finding LCA: general algorithm、multiplication algorithm、RMQ algorithm and Tarjan algorithm. Here we choose the **multiplication algorithm** for reference. In order to extract useful information from the file downloaded from database http://geneontology.org/ and turn GO terms' ID and their relationships into a plot, we adopt a method called **linked-forward-star** to save plot. Besides, as we all know, the structure of GO terms is definitely not a tree, that is a child node may has more than one parent nodes, resulting in one node with more than one depth. While in our package, we innovatively set every GO term's **deepest depth** as its real depth., so that the extract file contains more gene information and make it a tree structure. Additionally, to present the relationships between GO terms vividly and help to make our point better, we write a code in Python that enables to **visualize** the data set.

Perspective

Although our package enables to finding LCA between GO terms effectively and innovatively solves some problems existed, there are some simplification on data processing and limitation on application.

First, we simplify the relationships between GO terms. There are in fact three kinds of relationships : `is_a`, `part_of`, `` `regulates `` provided in data file.

```
A is_a B == A <-- B
A part_of B == A <-- B
A regulates B == A --> B
```

The `part_of` and `regulates` relations are a bit ambiguous, and their introduction will dramatically increase the complexity and difficulty of the algorithm. Besides, the relationship we emphasize here is strict parent and child relation, that is `is_a` , so we only extract `is_a` id as its parent node as simplification.

second, data size is limited. In our package, we set `N=300` (max number of GO terms). So before executing the package, it's better to make sure `N` is suitable for the data file and change its number if needed.

We will work on the problems mentioned above to improve our algorithm further. We anticipate that `"GoLCA.h"` will be useful and efficient for finding Lowest Common Ancestor(LCA) between Gene Ontology (GO) terms.