5P-02

Efficient Algorithm for Finding Diameter of Protein-Protein Interaction Networks

Hiroshi Sato† Yusuke Sano‡ Daiya Watari‡ Taku Ozaki‡ Katsuhisa Yamanaka‡ Takashi Hirayama‡

†Graduate School of Science and Engeneering, Iwate University ‡Iwate University

1 Introduction

Protein-protein interactions are physical contact between two or more protein molecules. Protein-protein interactions allow proteins to communicate with other proteins and change function of proteins. A protein- protein interaction network (PPI network) represents protein-protein interactions as a graph. In a PPI network, vertices are proteins and edges are protein-protein interactions. It is said that PPI networks satisfy small world and scale-free properties[1, 2].

In this paper, we first investigate the scale-free properties of some PPI networks published in Database of Interacting Proteins (DIP) [3]. Next, using the property, we propose an algorithm that computes the diameters of PPI networks efficiently.

2 Definition

This section gives definitions on graphs. Let $G = (V, E)$ be an undirected and unweighted graph, where V is a vertex set and E is an edge set. The *degree*, denoted by $d(v)$, of a vertex v is the number of the vertices adjacent to v. Let $P = \langle v_1, v_2, \ldots, v_k \rangle$ be the sequence of vertices of G. The sequence P is a *path* of G if P satisfies both (1) $(v_i, v_{i+1}) \in E, 1 \leq i \leq k$ and (2) $v_i \neq v_j, i \neq j, 1 \leq i, j \leq k$. The *length* of a path is the number of edges of P. The *distance*, denoted by $\mathsf{dist}_G(v_i, v_j)$, between two vertices v_i and v_j of G is the length of a shortest path between the two vertices. The *diameter*, denoted by $\mathsf{diam}(G)$, of G is the maximum distance in G. That is,

$$\mathsf{diam}(G) = \max\{\mathsf{dist}_G(v_i, v_j)) \mid v_i, v_j \in V\}.$$

3 Scale-free property of protein-protein interaction networks

In this section, we investigate the scale-free properties of some PPI networks published in DIP. We first explain the scale-free property of a graph.

Let $G = (V, E)$ be an undirected and unweighted graph, where $n = |V|$, and let n_k be the number of vertices of degree k in G. We define $P(k) := \frac{n_k}{n}$. That is, $P(k)$ is the ratio of degree-k vertices in G. A graph is *scale-free* if the degree distribution of the graph almost follow the power law of negative exponent, and it holds

$$P(k) \propto k^{-\gamma}.$$

It is known that the log-log graphs of degree distributions of scale-free networks forms almost a downward-trend straight line.

Now, we describe our results. We investigate the scale-free property of some PPI networks published in DIP. In DIP, PPI networks based on experimental results of protein-protein interaction are published. We investigated the degree distributions of the following 11 PPI networks: Li 2004a, Giot2003a, Gavin2002a, Dmela20170205, Scere 20170205, Ecoli20170205, Celeg20170205, Hsapi20170205, Hpylo20170205, Mmusc20170205, Rnorv20170205. Figure 3 shows the degree distribution of one of the 11 PPI networks. As we can see the graph forms almost a downward-trend straight line. Due to space limitation, in this report, we omit the distributions of the other PPI networks. However, we confirmed that all the distributions follow the power law of negative exponent.

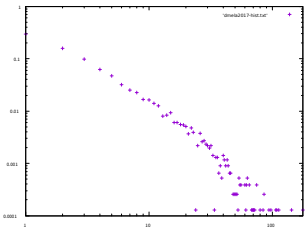

Figure 1: Degree distribution of dmela20170205 (Drosophila melanogaster).

4 Computation of diameter algorithm

In this section, we propose an algorithm for computing the diameter of a graph. A diameter is one of the most fundamental parameters of graphs. We first describe a naive method using Warshall-Floyd's algorithm. Then, we improve the method using the property of PPI networks.

4.1 Computing diameter using Warshall-Floyd's algorithm

Warshall-Floyd's algorithm is a well-known algorithm that computes the distances of all vertex pairs in $O(n^3)$ time. We first apply Warshall-Floyd's algorithm, then just output the maximum distance as the diameter of an input graph. Warshall-Floyd's algorithm is well-known, however, for self-containment, we show a pseudo-code of the algorithm in **Algorithm 1**. Here, $\texttt{temp\_d}_G[i][j]$ represents a temporary distance between two vertices v_i and v_j of G. At the end of the algorithm $\texttt{temp\_d}_G[i][j]$ takes the distance between v_i and v_j of G.

Algorithm 1: Diam-Warshall-Floyd(G)

```
if (v_i, v_j) ∈ E then
    temp_d_G[i][j] = 1
else
    temp_d_G[i][j] = ∞
for k ← 1 to n do
    for j ← 1 to n do
        for i ← 1 to n do
            temp_d_G[i][j] ←
              min(temp_d_G[i][j], temp_d_G[i][k] +
              temp_d_G[k][j])
Output max{temp_d_G[i][j] | 1 ≤ i, j ≤ n}
```

4.2 Proposed algorithm

In this subsection, we improve **Algorithm 1** for PPI networks. If a graph has one or more edge, isolated vertices can be ignored when we compute the diameter of a graph. To make discussion simple, we assume that an input graph has no isolated vertex.

We introduce some notations. Let $G = (V, E)$ be an undirected and unweighted graph, and let $V_1 \subseteq V$ be a set of the vertices of degree 1 in V. We denote by $V_2 = V \setminus V_1$ the set of the vertices except V_1 and by G_2 the graph induced by V_2.

Now, we explain the idea of our algorithm. As we see in the previous section, PPI networks have scale-free property. Scale-free networks have a lot of vertices of small degrees, especially vertices of degree 1. Our algorithm focuses on this property. First, we compute $\mathsf{diam}(G_2)$ using **Algorithm 1**. Then, we compute $\mathsf{diam}(G)$ from $\mathsf{diam}(G_2)$ taking degree-1 vertices into consideration.

We explain the details of our algorithm. The algorithm first computes $\mathsf{diam}(G_2)$ using **Algorithm 1**. We define L_1 and L_2 as follows:

$$L_1 := \{\{v_i, v_j\} \mid v_i, v_j \in V_2, \mathsf{dist}_{G_2}(v_i, v_j) = \mathsf{diam}(G_2)\}$$
$$L_2 := \{\{v_i, v_j\} \mid v_i, v_j \in V_2, \mathsf{dist}_{G_2}(v_i, v_j) = \mathsf{diam}(G_2) - 1\}$$

Then, we can compute $\mathsf{diam}(G)$ as follows. If there exists a pair $(v_i, v_j) \in L_1$ such that both v_i and v_j are adjacent to degree-1 vertices in G, then it holds $\mathsf{diam}(G) = \mathsf{diam}(G_2) + 2$. Now, let us consider the case that no such vertex pair in L_1. If there exists a pair $(v_i, v_j) \in L_1$ such that either v_i or v_j is adjacent to degree-1 vertex in G, then it holds $\mathsf{diam}(G) = \mathsf{diam}(G_2) + 1$. Next, let us consider the case not the above two cases. If there exists a pair $(v_i, v_j) \in L_2$ such that both v_i and v_j are adjacent to degree-1 vertices in G, then it holds $\mathsf{diam}(G) = \mathsf{diam}(G_2) + 1$. We assume that all of the above conditions do not hold. Then, it holds $\mathsf{diam}(G) = \mathsf{diam}(G_2)$. The pseudo-code of our algorithm is shown in **Algorithm 2**. The running time is $O(n^3)$, which is the same as **Algorithm 1**.

Algorithm 2: Proposed Algorithm(G)

```
Construct G_2 from G
diam(G_2)=Diam-Warshall-Floyd(G_2)
L_1 = {{v_i, v_j} | v_i, v_j ∈ V_2, dist_{G_2}(v_i, v_j) = diam(G_2)}
L_2 = {{v_i, v_j} | v_i, v_j ∈ V_2, dist_{G_2}(v_i, v_j) =
  diam(G_2) − 1}
if there exists {v_i, v_j} ∈ L_1 such that both v_i, v_j are
  adjacent to vertices in V_1 in G then
    Output diam(G_2) + 2
else if there exists {v_i, v_j} ∈ L_1 such that either v_i or
  v_j is adjacent to a vertex in V_1 in G then
    Output diam(G_2) + 1
else if there exist (v_i, v_j) ∈ L_2 such that both v_i and
  v_j are adjacent to vertices in V_1 in G then
    Output diam(G_2) + 1
else
    Output diam(G_2)
```

Table 1: Comparison of running time.

Data names	Diam-Warshall-Floyd (s)	Ours (s)
Giot2003a	21102.67	7376.39
dmela2017	32040.18	10450.92
hpylo2017	29.29	5.31
Gavin2002	140.57	37.78
Li2004a	966.36	77.02
celeg2017	1085.76	86.96
ecole2017	1407.23	354.74
hsapi2017	5697.60	795.06
mmusc2017	468.15	27.87
rnorv2017	9.02	0.37
scere2017	10094.69	4528.50

5 Experimental results

Table 1 shows experimental results. We compared two algorithms: **Algorithm 1** and **Algorithm 2**. Environment of experiment is as follows. Programming language: Common lisp, Compiler: SBCL1.3.1.debian CPU: AMD FX(tm)-8350 Eight-Core Processor, and Memory: 16GB. From the experimental results, we can see that our algorithm is faster than the naive one although the running time of both algorithms is $O(n^3)$.

References

[1] Duncan J.Watts and Steven H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature International Journal of Science*, Vol. 393, pp. 440–442, 1998.

[2] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, Vol. 286, pp. 509–512, 1999.

[3] Database of Interacting Proteins. https://dip.doe-mbi.ucla.edu/dip/Download.cgi.