Small World Phenomena and the Greedy Algorithm

Chenkai Wang

Abstract

In this essay, we first introduce the small world phenomena. After that, we use the package "networkx" [1] and the greedy algorithm to find the path. Finally, the simulation is applied to reveal the relationship between the algorithm time consumption and the clustering coefficient.

Keywords: Small World Phenomena, Greedy Algorithm, Simulation

Contents

1 Introduction of the problem

1.1 Small World Phenomena

In 1967, Milgram completed his famous sociological small world experiment[3], revealing that we are much closer to a stranger than we think. More specifically, randomly take two people on the planet, averagely we only need 5.5 mediators to get in touch with each other. The picture below gives an example of a possible path.

Figure 1: One possible path in the small world experiment

In conclusion, short paths are ubiquitous in the real world. Then a natural question arises: how can we find such paths? In section 2, we will provide an algorithm to solve it.

1.2 From the Reality to the Network

Think of every person on the planet as a node. If two people know each other, use a bilateral edge to connect the nodes of those two people so that we can get a social network. A natural thought is that once we can fully build such a network, we might be able to pinpoint the distance between any two people. However, building such an explicit network is impossible. There are at least three reasons:

- store a network with so many nodes and edges is expensive;
- people are socializing and making new friends any time and any where, and the network cannot give real-time feedback;
- we do not care about the exact distance of two picked people. Instead, we focus more on the average distance between two people and the existing path.

Therefore, we need to build a network that has the following features. On the one hand, its construction rules should be as simple as possible, which makes it convenient for us to study the behavior of different scales. On the other hand, the simple structure network can be combined with the small world phenomenon, which means existing a solid interpretation.

2 The construction of the Network and the Greedy Algorithm

2.1 The construction of the Network

We derives a network below from an $n \times n$ lattice with two kinds of edges to meet the requirement.

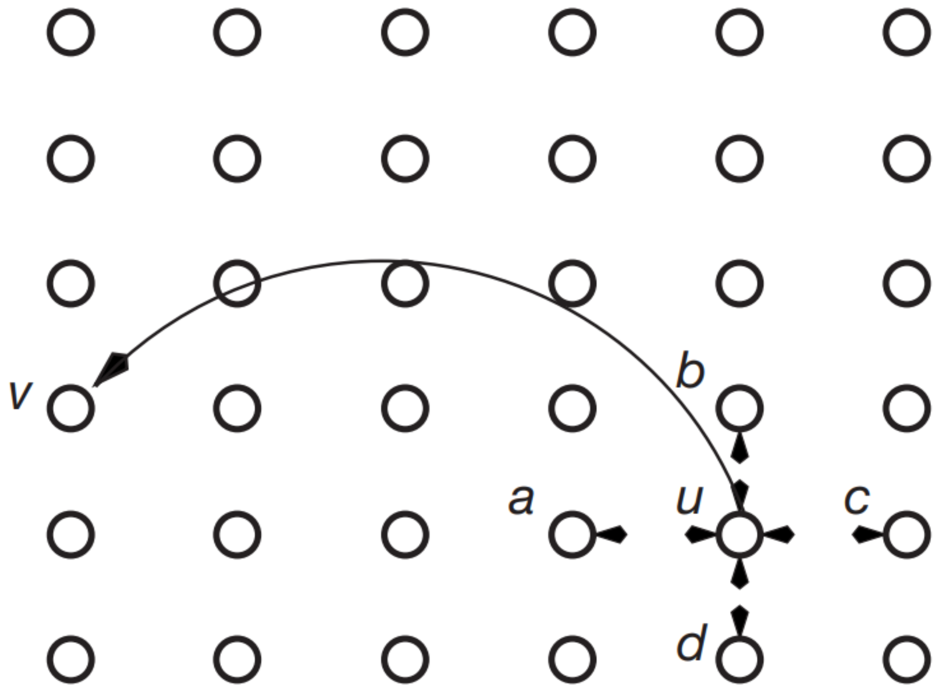

Figure 2: The built network[2]

- The former we call it directed short-range edge. For each node, say u, has a short-range connection to its nearest neighbors, say a, b, c, d.
- The latter we call it directed long-range edge. For each node, say u, has a probability proportional to $d(u, v)^{-r}$ to be connected to a randomly chosen node v, where r is the fixed clustering coefficient and $d(u, v)$ denotes the Manhattan distance of u and v.

To continue, we need to provide a more precise definition. In the above $n \times n$ network $\{(i, j) : i \in \{1, 2, \ldots, n\}, j \in \{1, 2, \ldots, n\}\}$, the distance between any two nodes $u(i, j)$ and $v(k, l)$ is defined as $d(u, v) = |k - i| + |l - j|$. Moreover, we introduce two universal parameters, p, and q. The node u has a directed short-range edge to its nearest neighbors with distance p. In figure 2, $p = 1$. Besides that, The nodes u will have q directed long-range edges. In figure 2, $q = 1$.

Now the problem becomes that for the selected target node v, how to find the shortest path from the specified node u. We will propose a greedy algorithm to solve the question in next section.

2.2 the Greedy Algorithm

Algorithm 1 The greedy algorithm

Input: n, p, q, r, u, v

Create ω and S_{temp}.
while $u \neq v$ **do**
    Find the short-range connections to $u's$ nearest neighbors according to parameter p and add the nodes to the set S_{temp}.
    Find the long-range connections to u according to parameter q and add the nodes to the set S_{temp} again.
    $\omega \leftarrow$ *the node in* S_{temp} *with smallest distance to* v.
    if $\omega \neq v$ **then**
        Add ω to S.
        $u \leftarrow \omega$.
        clear S_{temp}.
    end if
end while

Output: S

The input parameters have already been explained in section 2.1. Notice that u always gets closer to v after each circulation we know that the algorithm must converge eventually. We know that the greedy algorithm obtains an optimal local solution, but the locally optimal solution is not always the optimal global solution. However, here we do not care whether we get global optimal or not due to the following reasons:

- it is almost impossible to get a global optimal;
- even if we could get a global optimal, it is not affordable;
- we get local optimal with very little time and space complexity compared to global optimal, and we are happy to lose a little accuracy in exchange for a huge increase in efficiency.

In next section, we will find the relationship between the time cost of the algorithm and the clustering coefficient.

3 Simulation of the Greedy Algorithm

The primary figure of merit is its expected cost time T, which represents the expected number of steps from the specified node u to the target node v. The greedy algorithm use only local information. Compared with a global knowledge of all connections in the network, the shortest path can be found very simply.

Now we set $n = 20000$, $u = (1, 1)$, $v = (20000, 20000)$, $p = 2$, $q = 2$. We want to find the relationship between expected cost time T and clustering coefficient r. Let r increase from 0 by 0.1 to 2.5 and record the cost time T of each r. We can get the following picture.

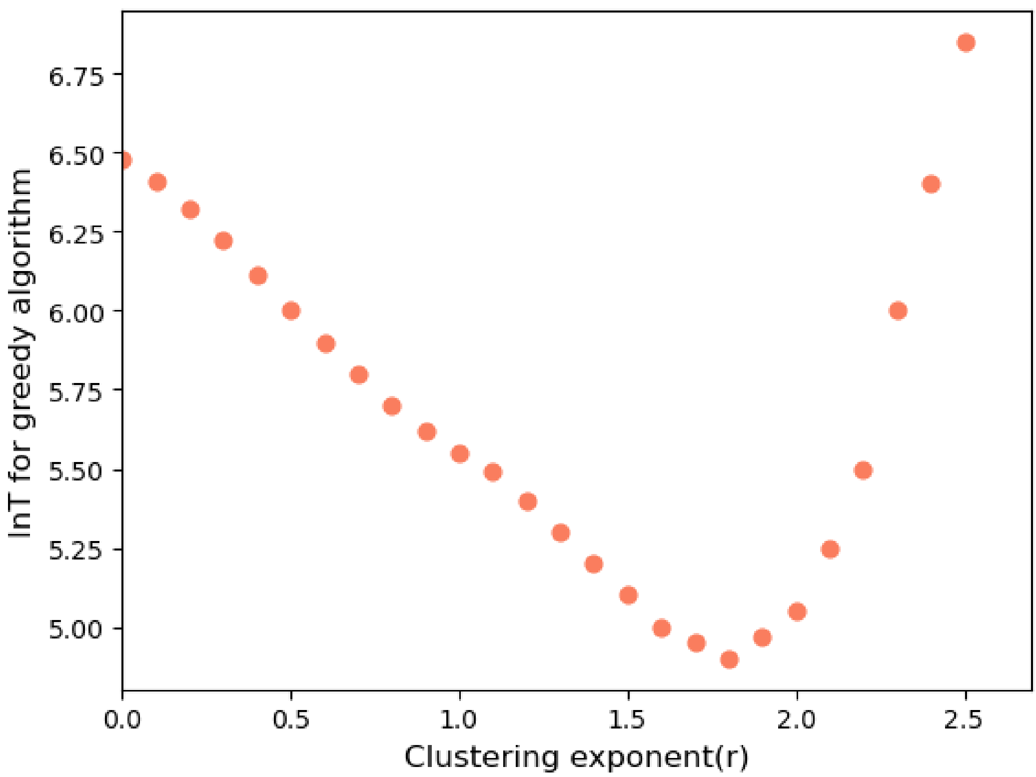

Figure 3: The relationship between clustering exponent $r$and lnT

References

[1] Dan Schult et. al Aric Hagberg Pieter Swart. https://networkx.org.

[2] Jon M Kleinberg. "Navigation in a small world". In: *Nature* 406.6798 (2000), pp. 845–845.

[3] Stanley Milgram. "The small world problem". In: *Psychology today* 2.1 (1967), pp. 60–67.