Information Sciences 422 (2018) 282–289

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Converting a network into a small-world network: Fast algorithms for minimizing average path length through link addition

Andrew Gozzard, Max Ward, Amitava Datta*

School of Computer Science and Software Engineering University of Western Australia, Perth WA 6009, Australia

ARTICLE INFO

Article history:
Received 22 December 2016
Revised 1 September 2017
Accepted 5 September 2017
Available online 8 September 2017

ABSTRACT

The average path length in a network is an important parameter for measuring the end-to-end delay for message delivery. The delay between an arbitrary pair of nodes is smaller if the average path length is low. It is possible to reduce the average path length of a network by adding one or more additional links between pairs of nodes. However, a naïve algorithm is often very expensive for determining which additional link can reduce the average path length in a network the most. In this paper, we present two efficient algorithms to minimize the average network path length by link addition. Our algorithms can process significantly larger networks compared to the naïve algorithm. We present simple implementations of our algorithms, as well as performance studies.

1. Introduction

Many real-world networks are modelled as a graph, $G = (V, E)$, where V is a set of vertices and E is a set of edges connecting some of the vertex pairs from the set V. Any message sent from a source to a destination propagates through intermediate vertices. A packet incurs *end-to-end delay* if it has to go through many intermediate hops. Since any pair of nodes can act as a source-destination pair, it is desirable that all paths in a network go through as few intermediate hops as possible. In other words, any traffic in the network will incur less average delay if the average path length in the network is low. Quite often the edges of these graphs are considered to be weighted. The weights may indicate the cost of establishing an edge, the latency or other factors depending on the context of the network.

Many networks have regular structures, meaning each node is connected to an equal number of nodes on an average. This makes the average path lengths in real-world networks quite long. In *small-world* networks the degree distribution of the nodes follows a power law and the average distance between any pair of nodes is usually small. These networks are of interest for quite sometime since Milgram's pioneering paper [9]. Watts and Strogatz [14] designed an approach that improves the clustering coefficient of a random network and converts a random network into a small-world network. Comellas and Sampels [1] replaced each node of a network by a mesh network, the number of nodes in the mesh is equal to the degree of the node that is replaced by the mesh. This converted an arbitrary network into a small-world network. Fall [5] showed that the addition of a few random edges in a network can reduce the average path length of a network

* Corresponding author.
E-mail address: amitava.datta@uwa.edu.au (A. Datta).

http://dx.doi.org/10.1016/j.ins.2017.09.020
0020-0255/

significantly. Lu et al. [15] showed that an arbitrary network can be made into a small-world network by introducing a scale-free distribution of nodes in a binary tree network.

The small-world nature of a network can be characterised in several different ways. However, our focus in this article is the average path length for all pairs of nodes. Usually a lower average path length reduces the overall communication cost in a network. Though there is strong evidence from previous work by Fall [5] that addition of extra edges can reduce the average path length in a network, the algorithmic nature of this problem has been studied only recently.

Meyerson and Tagiku [10] have shown that the general problem of adding k edges for minimizing the weighted average shortest path lengths between all pairs of vertices is NP-hard. They designed several approximation algorithms for this problem [10]. We are interested in this paper in designing an algorithm that adds a single edge to an existing network for minimizing the average shortest path lengths between every pair of vertices. Though it may seem quite a restricted problem, it has applications in many different areas. Chang et al. [3,4] have considered the problem of adding radio-frequency (RF) links in multi-core processor design. Their aim is to reduce the latency of communication in multi-core architectures, by adding extra radio frequency links on top of a regular interconnection scheme like a mesh network. However, RF interconnects require much more area and cannot replace the traditional interconnects. Adding even a single RF interconnect can improve latency significantly [3,4]. The weight on the link is the area requirement for an RF interconnect in this case. Ogras and Marculescu [12] consider a long range link over a regular mesh network for designing efficient Network-on-Chip (NoC) for VLSI design. The length of the long link is the weight on the link in this case. Pickavet and Demeester [13] have designed heuristic algorithms for link restoration in Synchronous Digital Hierarchy (SDH) networks. One of the key parts in their four-phase algorithm is local optimization, where they try to improve the topology in terms of the spare capacity, by adding an extra link. The weight on the link is the spare capacity in this case. Jin et al. [8] and Newman [11] have simulated stronger community structures in artificial social networks by link addition. It is possible to strengthen a community by adding an extra link between two nodes v_i and v_j such that they share a relatively large number of friends. The probability of a link addition is the weight on an edge in this case and the probability is determined by counting the number of common friends.

Recently Gaur et al. [7] have studied several deterministic link addition strategies for converting an arbitrary unweighted and undirected network into a small-world network. They found that the best strategy is to add an additional (long) link in an arbitrary network to minimize the average path length of the network. Given a graph $G = (V, E)$, Gaur et al. [7] considers an $O(V^2 \log V)$ algorithm for the all-pairs shortest path algorithm for unweighted and undirected graphs. This algorithm is not an original contribution by Gaur et al. as they have stated explicitly in their paper [7]. There is a possibility of introducing $O(V^2)$ new edges in a connected graph. The all-pairs shortest path algorithm is run after introducing each new edge and the average path length is computed. Hence the all-pairs shortest path algorithm needs to be run $O(V^2)$ times and the overall complexity of determining which link addition results in the minimum average path length is $O(V^2 \times V^2 \log V) = O(V^4 \log V)$.

In this paper we present two efficient algorithms for the problem of introducing a new link for minimizing the average path length. Our algorithms, unlike that considered by Gaur et al. [7], works on weighted, directed graphs, for both positive and negative weights. Hence the complexity of our algorithm is asymptotically strictly better than the algorithm considered by Gaur et al. [7]. We present our algorithms, time complexities and also performance results in this paper. Our first algorithm runs in $O(V^4)$ time in the worst case, and our second algorithm runs in $O(V^3 \log V)$ time in the worst case, as opposed to the naïve algorithm that takes $O(V^5)$ time for weighted directed graphs.

2. Efficient algorithms for link addition that minimizes the average path length

Following standard convention, we do not consider graphs with self-loops, negative weight cycles and multiple edges between same pair of vertices. We define a graph as $G = (V, E)$, where V is the set of vertices and E is the set of edges. S is the set of edges that can be added to the graph. We call S as the set of *augmenting* edges. We slightly abuse notations to denote these sets and their cardinalities using the same symbols. In general, the shortest path between vertices v_i and v_j is the path that has the minimum sum weight. In the simplest case each edge has a unit weight, or is unweighted, and the shortest path is the minimum number of edges between v_i and v_j. The *single-source shortest path* (SSSP) algorithm finds the shortest paths between a given vertex v_i and all other vertices in the graph; and the *all-pairs shortest path* (APSP) algorithm finds the shortest paths between all pairs of vertices. We refer the reader to the book by Cormen et al. [2] for a review of shortest path algorithms. We will use the Floyd–Warshall algorithm as the basis of our new algorithm, as it is the best known algorithm for dense graphs [2]. We refer to the problem of *link addition that minimizes the average path length* as the *MinAPL* problem as in [7]. A naïve application of the Floyd–Warshall algorithm can solve this problem in the following way. We first compute the APSP of a graph G by executing the Floyd–Warshall algorithm in $O(V^3)$ time, and compute the average path length of G. Next, we introduce all possible $O(S)$ additional links and compute the average path length of the resulting graph by using the Floyd–Warshall algorithm and determine the link whose addition minimizes the average path length. This algorithm will take $O(S \times V^3) = O(V^5)$ time in the worst case, as $S = O(V^2)$ in the worst case. We present two algorithms in this paper that improve upon this time complexity and facilitates the processing of much larger graphs for link addition. Our algorithms are based on some common observations and have varying performance depending on the sparsity of the input graph. We call these two algorithms as *FirstMinAPL* and *SecondMinAPL*.