Shortest Path Routing Algorithms in Multihop Networks

Sudip Misra
Cornell University, USA

INTRODUCTION

The topic of *shortest path algorithms* is very fundamental and important in information science and technology. Shortest path algorithms have evolved over many years and have found applications in different domains such as telecommunication networks, military, and transportation. There has been a lot of work undertaken on this topic in this area in the past. A lot of research is still being conducted. The topic is still poorly understood. This article should be helpful to readers, because it reviews some of the important works conducted in the area, out of the plenty of works available on this topic. The problem is so fundamental that whatever the interest areas of the readers may be, they will find the article useful.

The popular traditional shortest path algorithms date back to 1958/1959 and were proposed by Dijkstra (1959), and Bellman (1958). Their algorithms found wide applications in the abovementioned domains for many years. However, they were static. Thereafter, many other algorithms were proposed in the last few decades, all of which can be classified to be either dynamic or semi-dynamic.

BACKGROUND

Multihop networks, such as the Internet and *mobile ad hoc networks* (MANETs) contain several routers and mobile hosts. The Internet typically employs routing protocols such as the open shortest path protocol (OSPF) and the intermediate system—intermediate system protocol (IS-IS), and the MANETs employ protocols such as the fisheye state routing (FSR), the optimized link state routing (OLSR), and the ad hoc on-demand distance vector routing (AODV).

In many of these protocols, each router (or a routing device) computes and stores a shortest path tree (SPT) from one router to all other routers and hosts in a routing domain (Moy, 1997; Peterson & Davie, 2000; Schwartz & Stern, 1980). Such networks, which can be modeled as graphs (Misra & Oommen, 2005b; Ramalingam & Reps, 1996), typically contain several routers/switches (nodes) connected by links (edges) with constantly changing costs (weights), link-ups (edge-insertions), and link-downs (edge-deletions).

SINGLE-SOURCE SHORTEST PATH ROUTING: DYNAMIC VERSUS STATIC

The problem of computing and maintaining information about the shortest paths information in a graph (with a single source)—where the edges are inserted/deleted and edge-weights constantly increase/decrease—is referred to as the *dynamic single source shortest path problem* (DSSSP). Although this problem is important, it has received little attention in the literature. The importance of the problem lies in the fact that it is representative of many practical situations in daily life, where most environments are dynamically changing. In such environments, one needs to devise efficient solutions to maintain the shortest path even though there are updates on the structure of the graph by virtue of edge-insertion/deletion, or edge-weight increase/decrease, and hopefully this can be achieved without recomputing everything “from scratch” following each topology update. An example of a single-source shortest path graph, after the insertion of an edge is shown in Figure 1. The new edge C→F appears in the list of shortest paths, and the existing edge B→F, which was earlier in the list of shortest paths, ceases to be so.

Out of the four possible edge operations (insertion/deletion and increase/decrease), it has been shown that edge-insertion is equivalent to edge-weight decrease, and edge-deletion is equivalent to edge-weight increase. Increasing or decreasing an edge-weight can be performed by inserting a new edge (with the new weight) parallel to the edge under consideration, and then deleting the old edge (Ramalingam & Reps, 1996). If all edge operations are allowed, the problem is referred to as the *fully dynamic problem*. If only edge-insertion/weight-decrease (or edge-deletion/weight-increase) is allowed, the problem is referred to as the *semi-dynamic problem* (Frigioni, Marchetti-Spaccamela, & Nanni, 1996).

Typically, with many present-day routing protocols,[1] with a unit change in network topology (e.g., link-ups, link-downs, and link-cost changes), each router in the routing domain is intimated of the change. This change typically triggers recomputation of each router’s SPT.

There are well-known *static solutions* to the traditional combinatorial *single source shortest path problem* (Bellman, 1958; Dijkstra, 1959) which are unacceptably inefficient in such dynamic practical scenarios because using them would involve recomputing the shortest path tree from scratch each time a topology change occurs in the graph. Static algorithms are unarguably more effective in fixed-infrastructure networks because of their polynomial time complexity. But they are extremely inefficient for time-critical, rapidly changing infrastructures. For instance, if such an algorithm is used in a real-time, large network routing scenario, where there are fast link-state changes, such recomputations will delay the execution of important routing functions considerably.

Two of the earliest known works on the dynamic shortest path problem date back to the papers by Spira and Pan (1975) and McQuillan, Richer, and Rosen (1980). While the former is theoretically proven to be inefficient, the latter has neither been proven theoretically nor through simulations.

The most recent and well-known solutions to the DSSSP problem on general graphs with positive real-valued edge-weights were proposed by Ramalingam and Reps (1996), Franciosa, Frigioni, and Giaccio (1997), and Frigioni, Marchetti-Spaccamela, and Nanni (2000). However, the solution by Franciosa et al. (2000) is limited to the semi-dynamic problem only. Although these recent works are theoretical in nature, Ramalingam and Reps (1996)'s and Frigioni et al. (2000)'s results were recently experimentally evaluated through simulations (Demetrescu, Frigioni, Marchetti-Spaccamela, & Nanni, 2001; Frigioni, Ioffreda, & Nanni, 1998). While the former was found to be superior when it concerns running time, the latter was shown to be better suited when the worst-case time was the main concern, or when the number of edges to be updated had to be minimized.

The well-known, fully dynamic algorithms (mentioned previously) are constrained by the following limitations:

1. The existing fully dynamic algorithms process unit changes to topology (i.e., edge-insertion/deletion or weight-increase/decrease) one change at a time, that is, sequentially. When there are several such operations occurring in the environment simultaneously, the algorithms are quite inefficient.
2. In environments where the edge-weights change stochastically and continuously, the existing algorithms (mentioned previously) would fail to converge to the actual underlying "average" solution.

The problems are worse in large topologies which have a large number of nodes and edges, and where a large number of topology changes can occur continuously at all times. In such cases the existing algorithms would fail to determine the shortest path information in a time-critical manner.

Since such scenarios are representative of the actual environments in which the dynamic shortest path algorithms are likely to operate, the existing solutions would be limitedly useful. Misra and Oommen (2005b) proposed a learning solution by taking the aforementioned aspects into

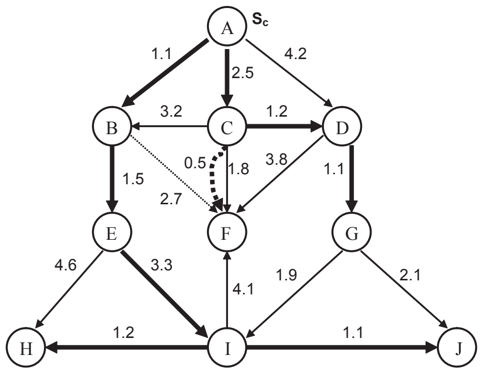

Figure 1. Graph after the insertion of the edge C→F with weight 0.5

consideration. To the best of my knowledge, other than their solution, there is no known solution to finding the shortest path in a real-weighted graph where multiple edges are changing stochastically at once, and at the same time, which is more efficient than calculating everything from scratch for every change. The work reported by them was inspired by the need for formulating an algorithm for finding the shortest path in such realistically occurring stochastic environments. Indeed, they sought to find the shortest path in the "average" graph (dictated by an "Oracle," also called the *environment*). Since, on query, the edge-weights supplied by the environment are assumed to follow an underlying unknown distribution, there exists a mean solution to the problem to which the algorithm would converge to after a sufficiently long time. Their intention was to find the "statistical" shortest path in the average graph that will be stable—regardless of the (possibly) continuously changing weights provided by the environment. Their solution generates superior results (when compared to the previous solutions). However, unfortunately, their scheme does not consider the insertion/deletion of edges. Thus, the problem they have considered assumes that there is one fixed structure graph with randomly changing edge weights, and that the distribution of these random variables is unknown.

ALL-PAIRS SHORTEST PATH ROUTING: DYNAMIC VERSUS STATIC

Contrary to the previous discussions, in which the idea was of computing the shortest paths from one node to all the other nodes in a network, the problem of computing and maintaining all-pairs shortest paths information in a graph where the edges are inserted/deleted and edge-weights constantly increase/decrease is referred to as the *Dynamic*

Related Content