An Efficient Algorithm for the Shortest Path Problem with Forbidden Paths

Chiun-Chieh Hsu[1], Da-Ren Chen[2], and Hua-Yuan Ding[1]

[1] Department of Information Management, National Taiwan University of Science and Technology, 43, Sec.4, Keelung Rd., Taipei, 106, Taiwan, R.O.C.
cchsu@cs.ntust.edu.tw
[2] Department of Information Management, Hwa Hsia Institute of Technology, 111, Gong Jhuan Rd., Chung Ho, Taipei, 235, Taiwan, R.O.C.
danny@cc.hwh.edu.tw

Abstract. In this paper, we discuss the shortest path problem with forbidden paths (SPPFP), where the constraints come from a set of forbidden paths (arc sequences) that cannot be part of any feasible solution. SPPFP can be used to solve the problems with hard-to-modeled path constraints or to implement exact branching scheme.The method proposed by Villeneuve and Desaulniers first transfer the SPPFP problem to the *k*-shortest paths problem. We propose an algorithm which reduces execution time of solving the SPPFP problem via a nodes combination process. It improves the approach proposed by Villeneuve and Desaulniers. In addition, their algorithm for generating augmented paths could cause the inconsistency in particular situations. The proposed method also improves the weakness without increasing its time complexity.

Keywords: Shortest paths, Forbidden paths, Labeling algorithm, Network flows.

1 Introduction

The shortest path problem in the field of graph theories is an important concept. Sometimes errors on nodes among the network may prevent the best solution from deriving out. Therefore, the researchers start to focus on k-shortest paths problem. Its applications include an actual path for desired output, problems of scheduling, optimization problems, and so on. In recent years some researchers apply this problem on particular networks, like time windows network [5, 19, 20]. In the unrestricted problem, the paths are allowed to be looping. Several techniques, such as dynamic programming or sophisticated data structures, have been applied to this problem [8, 13, 14]. The restricted problem where only loopless paths are accepted is more difficult to solve. Most of the proposed methods for solving restricted problems are based on the branching approach [15, 21, 22].

The proposed algorithm reduces the execution time for solving the SPPFP problem by a *nodes combination* method. It also fixes the weakness of Villeneuve and Desaulniers' approach without increasing its time complexity. This paper is organized as follows. Related works are introduced in Section 2. The proposed algorithm is

A. Hua and S.-L. Chang (Eds.): ICA3PP 2009, LNCS 5574, pp. 638–650, 2009.

described in Section 3. Computational experiments comparing the behaviours of the classical and the new methods are reported in Section 4. Conclusions are presented in Section 5.

2 Related Work

This section introduces the concept of two algorithms playing a central role in our SPPFP approach. The first algorithm is k-shortest paths algorithm proposed by Martins [17], which could only extend/forbid one deviation/shortest path at once. The second is SPPFP algorithm proposed by Villeneuve and Desaulniers [18] using deterministic finite automaton [1], which could consider many forbidden paths in parallel.

Let G=(N, A) be a directed graph where N and A denotes its node and arc sets with n nodes and m arcs, respectively. Let s denotes a source node, t a sink node, and F a set of (forbidden) in G. The goal is to find shortest paths from nodes s to node t under the constraint that no paths in the solution contain any path $f \in F$ as a subpath.

2.1 Martins' Algorithm

Martins' k-shortest path algorithm is based on a simple concept: once a shortest path is determined, it duplicates nodes and arcs along the shortest path and removes the arcs to prevent the duplicated path from completing the entire forbidden path. These results in an enlarged network where all the paths except the deleted one can be determined. Let G=G^1 denotes the original graph, and the first shortest path is computed on G^1 by applying a classic shortest path algorithm. The ith shortest path, i=2, …, k, is computed by the same algorithm on a graph G^i, which is based on G^{i-1} and create a deviation paths from the (i-1)th shortest path. We define strict prefix of a given path is any sub-prefex except the empty path and the entire path itself, and x denotes the last node of a sequence x. There exists an arc (π, ρ) if and only if ρ corresponds to the longest strict prefix that can be obtained by extending the prefix corresponding to π with ρ. The following steps applied to G^{i-1} until the ith shortest path has been completed.

- Duplicate nodes (except nodes s and t) and outgoing arcs along the (i-1)th shortest path.
- Remove the first arc of (i-1)th shortest paths in G^{i-1}.
- Remove the the last arc in the newly created paths linking to node t.

Hence, a new augmented network G_{k+1} = (N_k, A_k) is created.

Fig.1 illustrates an example to Martin's transformation. The upper region of the figure presents the original graph G=G^1 in which the shortest path is assumed to be the path <s, u, v, t>. The lower region of the figure presents graph G^2 derived from G^1. In graph G^2, the nodes and arcs in dashed lines are duplicated along the forbidden path <s, u, v, t>. Because the first arc of forbidden path has been removed, the path <s, u, v, t> cannot be completed in G^2. In addition, the arc between the last node of the augmented path (say v_2) and node t has been removed, because the path <s, u_2, v_2, t> does not allow to be the shortest path solution in G^2. A new possible shortest path <s, u_2, v_2, y, u, v, t> in G^2 corresponds to the non-simple path <s, u, v, y, u, v, t> in graph G.