Discrete Optimization 18 (2015) 217–251

Contents lists available at ScienceDirect

Discrete Optimization

www.elsevier.com/locate/disopt

Shortest path algorithms for functional environments☆

Louis Boguchwal

ARTICLE INFO

Article history:
Received 6 January 2015
Received in revised form 20 September 2015
Accepted 21 September 2015
Available online 11 November 2015

Keywords:
Shortest path algorithms
Combinatorial optimisation
Networks
Sequential decision-problems
Generalised flow

ABSTRACT

This research generalises classic shortest path algorithms to network environments in which arc-costs are governed by functions, rather than fixed weights. We show that the asymptotic efficiency of our algorithms is identical to their classic counterparts. Previous results, since Knuth in 1976, require several restrictive assumptions on the functions permitted in the network. In contrast, our algorithms require only monotonicity. We present examples illustrating that this is the largest class of functions to which classic algorithms can be generalised. Applications of this work include critical path extensions to solve sequential decision-problems, and generalised network flow with nonlinear gain functions.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Since their invention in the 1950s, shortest path algorithms have been applied to transportation, telecommunications, and sequencing problems [1]. The average person encounters these algorithms almost daily, when seeking directions on the internet or on GPS devices. Phone companies solve shortest path problems when routing data traffic between various geographical locations. Shipping companies solve shortest path problems to efficiently deliver goods. Shortest path problems also emerge in less traditional settings, such as project management and DNA sequencing [1].

Standard scheduling problems can be formulated as classic shortest and longest path problems to minimise overall project completion-times [2]. For example, the critical path method has become pervasive across time-dependent applications over the past fifty years, such as project-management and industrial engineering [3]. However, some decision-makers lose sight of their objectives. This paper will demonstrate how a decision-maker could apply extensions of classic shortest path algorithms to actually minimise or maximise a project's objective, z, by optimising the sequence of constituent tasks in the project.

This research generalises classic shortest path algorithms to network environments where the length of an arc is governed by a function. We refer to these general environments as *functional environments* throughout

☆ This work is based on research done for a master's thesis at the London School of Economics and Political Science.

E-mail address: Louis.Boguchwal@gmail.com.

http://dx.doi.org/10.1016/j.disopt.2015.09.006
1572-5286/© 2015 Elsevier B.V. All rights reserved.

this work. The cost of a path from a node i to another node j is in terms of some variable of interest z, where the cost from s to itself is initial value z_0. Arcs in the classic environment have static costs, whereas arcs in the functional environment have dynamic costs.

These costs are represented by functions $f : \mathbb{R} \to \mathbb{R}$, denoted as $f_{ij}(z)\ \forall (i, j) \in A$, where (i, j) is an arc in arc-set A. We signify the node-set by N. Formal notation is established in Section 1.2. Traversing an arc changes the value of variable z. If $z = z_i$ at node i, traversing arc (i, j) implies that

$$z_j = f_{ij}(z_i). \tag{1.1}$$

Thus, cost $z_j = f_{ij}(z)$ depends on cost z_i. Formally, we say that the cost of arc[1] (i, j) is *governed* by function $f_{ij}(z)$. But for brevity's sake, we say that "arc (i, j) is governed by $f_{ij}(z)$". This generalises the classic shortest path problem, where $f_{ij}(z) = c_{ij} + z$. The functional shortest path problem is to find a minimum cost path from some $s \in N$ to some $t \in N$ in this dynamic environment.

We develop several algorithms to solve the problem, dependent upon network structure and the classes of functions permitted. First, we solve the functional shortest path problem over acyclic networks in which arcs are governed by monotonically increasing functions. Next, we extend that algorithm to solve the problem over acyclic networks in which arcs are governed by monotonic functions. Then we extend the algorithm further to account for networks of arbitrary structure in which arcs are governed by monotonically increasing functions. Finally, we explore networks of arbitrary structure in which arcs are governed by monotonic functions.

This paper is organised as follows. We first explore applications of this work. Next, we provide an overview of classic shortest path algorithms and results. Then we formally introduce the functional environment. We extend the classic label-setting and label-correcting algorithms to solve the shortest and longest path problems in the above settings in Sections 3–6. Notably, our algorithms only require that the functions $f_{ij}(z)$ governing arcs $(i, j) \in A$ are monotonic, bereft of further restrictions. We also demonstrate that the computational complexity of the algorithms developed is no greater than, and in fact identical to, their classic counterparts. Finally, we discuss future research and conclude.

1.1. Applications

This research is motivated by sequential decision-problems and network flow problems. We briefly describe these connections here, and in greater detail in Sections 1.1.1 and 1.1.2, respectively. Each application is based on the meaning attributed to a network in the generalised functional environment.

This work was ultimately inspired by one-party sequential decision-problems. We do not consider strategic interactions, as in game theory. If other parties are involved in the decision-problem, we take their actions as given and perfectly predictable. We can represent the outcome associated with a decision with the functions $f_{ij}(z)$ on the arcs, given the state of our objective prior to executing the decision. For example, consider a company seeking to allocate its revenue with the objective of maximising its funds over time. If the company currently has $\$z$ in revenue and chooses decision (i, j), then we could model the resulting revenue as $f_{ij}(z)$. Using the functional environment, we can characterise the revenue of the company as decisions are executed.

In context of network flow, we can interpret the functions $f_{ij}(z)$ on arcs to illustrate violations of conservation-of-flow. For example, if we send z gallons of oil through a pipe from i to j, then the function $f_{ij}(z)$ represents the amount of oil actually reaching j. This interpretation can be applied to both generalised maximum flow and generalised minimum cost flow problems.

1.1.1. Extending critical path

This research provides a framework to solve sequential decision-problems, which are often solved using dynamic programming methods. Since our algorithms generalise classic shortest path methods, they

[1] Technically, this is the cost of the entire path from s to j.

Download English Version:

https://daneshyari.com/en/article/1141496

Download Persian Version:

https://daneshyari.com/article/1141496

Daneshyari.com