European Journal of Operational Research 244 (2015) 715–729

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Network construction problems with due dates

Igor Averbakh[a,*], Jordi Pereira[b]

[a] Department of Management, University of Toronto Scarborough 1265 Military Trail, Toronto, Ontario M1C 1A4, Canada
[b] Departamento de Ingeniera Industrial, Universidad Catolica del Norte, Av. Angamos 0610, Antofagasta, Chile

ARTICLE INFO

Article history:
Received 10 August 2013
Accepted 9 February 2015
Available online 14 February 2015

Keywords:
Scheduling
Network design
Network construction
Emergency restoration
Integrated network design and scheduling

ABSTRACT

A network needs to be constructed by a server (construction crew) that has a constant construction speed which is incomparably slower than the server's travel speed within the already constructed part of the network. A vertex is recovered when it becomes connected to the depot by an already constructed path. Due dates for recovery times are associated with vertices. The problem is to obtain a construction schedule that minimizes the maximum lateness of vertices, or the number of tardy vertices. We introduce these new problems, discuss their computational complexity, and present mixed-integer linear programming formulations, heuristics, a branch-and-bound algorithm, and results of computational experiments.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

We consider the type of network construction problems introduced in Averbakh (2012) and Averbakh and Pereira (2012). The base model used in Averbakh and Pereira (2012) and in this paper is as follows. A transportation network needs to be constructed by a server (construction crew) that is initially located at one of the vertices (the depot). The server can build edges of the network with a constant speed, and can travel within the already constructed part of the network with infinite speed (in practice, travel times are typically negligible with respect to construction times). It is required to develop a construction schedule (an order of building the edges) that minimizes some scheduling objective that is a function of the recovery times of vertices, where the *recovery time* of a vertex is the time when the vertex is reached by the server for the first time and becomes connected to the depot.

As discussed in Averbakh and Pereira (2012), this base model is applicable in a variety of application settings, most notably: (a) when an extension of an existing transportation network needs to be constructed (then, the depot represents the existing network), and (b) in emergency situations when some local part of a transportation network (e.g., subway system, road/railway network, mine system, etc.) has been damaged/destroyed by a disaster (e.g., flood, earthquake, hurricane, landslide, etc.) or terrorist activity that led to disconnection of some vertices/parts of the network from the rest of the network, and it is required to plan restoration of the damaged/destroyed edges to restore access to the disconnected vertices (then, the depot represents the survived main network). For a discussion of these applications, the reader is referred to Averbakh and Pereira (2012).

The specific problem studied in Averbakh and Pereira (2012) was to minimize the sum of the (weighted) recovery times of the vertices (the Flowtime Network Construction Problem). In this paper, we assume that vertices have due dates for recovery, and consider two problems: minimizing the maximum lateness of the vertices and minimizing the number of tardy vertices. These models are particularly important in emergency situations when some vertices are disconnected from the main network as a result of a disaster or terrorist activity, and people/equipment can be trapped there. A deadline or due date for recovery of a vertex may represent a self-sustainability limit of the vertex, i.e., a limit on the time that the vertex can remain disconnected from the rest of the network without risk to people/equipment trapped there. We introduce these problems, discuss their computational complexity, and present mixed-integer linear programming (MILP) formulations, heuristics, a branch-and-bound algorithm, and results of computational experiments based on randomly generated instances and instances generated from data on infrastructure restoration works after the 2010 Chilean earthquake (Chilean instances). The experiments show that our branch-and-bound algorithm outperforms CPLEX applied to available MILP formulations. For example, the branch-and-bound algorithm was able to solve all Chilean instances to proven optimality within seconds, while CPLEX applied to our MILP formulations ran out of memory for most of the Chilean instances.

* Corresponding author. Tel.: +416 287 7329; fax: +1 416 287 7392.
E-mail addresses: averbakh@utsc.utoronto.ca (I. Averbakh), jorgepereira@ucn.cl (J. Pereira).

http://dx.doi.org/10.1016/j.ejor.2015.02.014
0377-2217/© 2015 Elsevier B.V. All rights reserved.

We note that network construction problems such as in Averbakh (2012) and Averbakh and Pereira (2012) and in the present paper combine network design and scheduling elements. They also have some routing aspects, in the following sense: (a) The problems are the limiting case of the routing-scheduling problems where the travel speed of the server is finite but very large with respect to the construction speed; (b) at any instant, the only edges available for construction to a server are those edges that are adjacent to the current position of the server in the modified network where the already constructed parts are shrunk to points, and thus the problems can be viewed as routing-scheduling problems in a dynamically changing network. Property (b) is particularly important in multi-server/multi-depot settings such as those considered in Averbakh (2012); the polynomial algorithms for path networks in Averbakh (2012) heavily use this property.

Let us give a brief overview of related work where issues of scheduling construction/restoration activities in a network were studied. Guha, Moss, Naor, and Schieber (1999) considered the power outage recovery problem where it is required to restore connectivity between customer vertices and generator vertices in an electric power network when some relay vertices fail. The repair of the failed vertices occurs in stages where due to a budget constraint only a subset of failed vertices can be repaired in every stage, and the goal is to minimize the total weighted waiting time of disconnected customers. For this problem, Guha et al. (1999) presented a number of theoretical approximation results.

Averbakh (2012) showed that a very broad class of network construction problems with multiple servers and depots is solvable in polynomial time if the underlying network is a path.

Nurre, Cavdaroglu, Mitchell, Sharkey, and Wallace (2012) consider integrated network design and scheduling (INDS) problems where capacitated arcs are installed in a network to increase the maximum flow that can go through the network, and it is required to schedule the arcs' installation on a set of parallel identical work groups. The objective is to maximize the cumulative (over a finite time horizon) weighted flow through the network. An integer programming formulation, valid inequalities, and heuristic dispatching rules are proposed. Cavdaroglu, Hammel, Mitchell, Sharkey, and Wallace (2013) consider similar problems involving a set of interdependent networks. Nurre and Sharkey (2014) consider INDS problems where the performance of the network is evaluated based on such characteristics as the maximum flow value, minimum cost flow value, shortest path length (with single or multiple destinations), and the minimum spanning tree length. They prove that the considered problems are NP-hard, and present a heuristic algorithmic framework based on dispatching rules. The INDS problems considered in Nurre et al. (2012), Nurre and Sharkey (2014) and Cavdaroglu et al. (2013) also combine network design and scheduling decisions; they are general and can capture a variety of settings and objectives including some connectivity-based objectives, e.g., the sum of the weighted recovery times of vertices. However, the specific INDS models considered in Nurre et al. (2012), Nurre and Sharkey (2014) and Cavdaroglu et al. (2013) do not seem directly suitable for representing the objectives considered in the present paper, i.e., the maximum lateness or the number of tardy vertices, as they are based on the overall network characteristics (the maximum flow value, the minimum spanning tree length, etc.), rather than on the lateness of individual vertices with respect to their due dates. From the modeling perspective, a difference between the INDS problems from Nurre et al. (2012), Nurre and Sharkey (2014) and Cavdaroglu et al. (2013) and the network construction problems from Averbakh (2012), Averbakh and Pereira (2012) and the current paper is that the models from Nurre et al. (2012), Nurre and Sharkey (2014) and Cavdaroglu et al. (2013) do not assume that the servers (construction crews) can travel only in the already constructed parts of the network. This makes the solution spaces different. From the solution perspective, the difference is often not important in the single-server/single-depot case (Averbakh & Pereira, 2012), but is important in multiple-server/multiple-depot case as it affects optimal solutions and the optimal objective value (Averbakh, 2012).

Elgindy, Ernst, Baxter, Savelsbergh, and Kalinowski (2014) and Engel, Kalinowski, and Savelsbergh (2013) study incremental network design problems where at each stage one edge is installed, and the objective is to minimize the total length of the shortest path (Elgindy et al., 2014) or the minimum spanning tree (Engel et al., 2013) over all stages. Connectivity issues do not arise in these problems since in the initial network the vertices of interest are assumed to be already connected. Kalinowski, Matsypura, and Savelsbergh (2015) consider similar problems where the objective is to maximize the cumulative maximum flow in the network over all stages. This problem can be viewed as a special case of the problem considered in Nurre et al. (2012).

The structure of the paper is as follows. In Section 2, we formally introduce our problems. In Section 3, we study their computational complexity. In Section 4, we develop MILP formulations. In Section 5, we describe heuristics used for obtaining initial solutions, and develop lower bounds to be used in a branch-and-bound approach. In Section 6, we present some structural results, and in Section 7 we describe a branch-and-bound algorithm. Results of computational experiments are discussed in Section 8. Some conclusions are stated in Section 9.

2. Problem formulation

Let $G = (V, E)$ be a connected network with the set of vertices $V = \{1, 2, \dots, n\}$ and the set of undirected edges E. To avoid confusion, nodes of the network G will always be called vertices, reserving the term "nodes" to nodes of branch-and-bound search trees. For each edge $\{i, j\} \in E$, let $c_{ij} > 0$ be the length of the edge, $c_{ij} < +\infty$. The edges of the network represent connections (e.g., roads) that have to be constructed. A server (construction team) that is initially located at vertex 1 (the depot) can build edges with the speed of 1 unit of length per unit of time. The server can travel within the already constructed (connected) part of the network with infinite speed; that is, at any time, the server can immediately relocate to any point of the already constructed portion of the network. The server starts working at time 0. The instant C_i when the server reaches a vertex i for the first time is called the *recovery time* of the vertex; this is the instant when the vertex becomes connected to the depot. A due date d_i is associated with each vertex $i > 1$; vertex 1 (the depot) does not have a due date. We assume that vertices are indexed according to non-decreasing due dates, i.e. $d_2 \le d_3 \le \dots \le d_n$. Value $C_i - d_i$ is called the *lateness* of vertex i. A vertex with positive lateness is called a *tardy* vertex. Our goal is to find a possible schedule of constructing the edges so as to minimize a scheduling objective. This problem will be referred to as **Problem L** if the objective is to minimize the maximum lateness of the vertices, and **Problem T** if the objective is to minimize the number of tardy vertices. Let Z_L^* and Z_T^* be the optimum objective values for Problems L and T, respectively.

Edges constructed before all vertices have been recovered are called *essential*. Clearly, there is an optimal solution without preemption, where the server does not interrupt construction of an edge once it has been started. After a choice of essential edges has been made, the order of building other edges is not important. Observe that there is always an optimal solution where the essential edges form a spanning tree, which allows us to structure the set of feasible solutions as follows: *it is required to choose a spanning tree of essential edges and an optimal order of constructing the essential edges.*

We also discuss **Problem F** where it is required to find a solution that meets all due dates, or to show that no such solutions exist. This problem corresponds to the situation where the due dates are interpreted as strict deadlines. Clearly, to solve Problem F, it is sufficient to solve Problem L or T and check whether the optimal objective value is greater than 0. However, we will see that often Problem F is

Download English Version:

https://daneshyari.com/en/article/479572

Download Persian Version:

https://daneshyari.com/article/479572

Daneshyari.com