A LINEAR PROGRAMMING APPROACH TO THE OPTIMUM NETWORK ORIENTATION PROBLEM

E.L. JOHNSON
T.J. Watson Research Center, IBM Research Yorktown Heights, NY, USA

P. PIERONI
Mathematical Institute, University of Pisa, Via Buonarroti 2, 56100 Pisa, Italy

Received 15 July 1983
Revised manuscript received 7 January 1985

In this paper a procedure is described to orient arcs of a graph so as to minimize the sum of the distances between certain given source-sink pairs. This work is a substantial part of the more general problem of orienting a road network in such a way as to optimize a specified objective subject to the requirement that one can go from any point of the network to any other one. The proposed method, related to the ideas of sequential optimization [1, 2, 3], uses column-generation as well as row-generation.

Key words: Oriented Graph, Strongly Connected Graph, Conflict Arc, Column-Generation, Row-Generation.

Introduction

Let $G$ be a connected graph corresponding to a road network. Let $l_{ij}$ be a cost associated to arc $(i,j)$ and let a set of pairs of source-sink vertices in the graph be given. Our main purpose is to orient the arcs of $G$ in such a way that the sum of shortest distances from source to sink and return from sink to source for each pair is minimized. This problem can be formulated as a 0-1 integer program whose size is, in general, very large. In this paper a method is described to solve this problem. The idea is to solve a finite number of smaller-sized linear programs.

A solution procedure to orient arcs of a graph

In order to formulate the problem of finding optimal orientations in a graph, denote by $K$ the number of source-sink pairs. Index by $\kappa, \kappa = 1, \ldots, K$, the generic $(s_\kappa, d_\kappa)$ pair. For each pair $\kappa$, denote by $I_\kappa$ the index set of all the chains from source $s_\kappa$ to sink $d_\kappa$ and let $P_\kappa = \{p^*_i\}_{i \in I_\kappa}$ denote the set of the chains from source $s_\kappa$ to sink $d_\kappa$. Denote by $c^*_i, i \in I_\kappa$, the cost of chain $p^*_i$. Introduce, for each $\kappa$, a binary vector $X^\kappa = \{x^*_i\}_{i \in I_\kappa}$ which represents the problem variables, defined as
follows:

\[ x^\kappa_i = \begin{cases} 1 & \text{if chain } p^\kappa_i \text{ is the chosen chain for the } \kappa \text{-th pair,} \\ 0 & \text{otherwise.} \end{cases} \]

The problem can be formulated as an integral linear program as follows:

\[
\begin{align*}
\min & \quad \sum_{\kappa=1}^{K} \sum_{i \in I_\kappa} c_i^\kappa x_i^\kappa \\
\text{subject to} & \quad x_i^\kappa \in \{0, 1\}, \quad i \in I_\kappa, \quad \kappa = 1, \ldots, K, \\
& \quad \sum_{i \in I_\kappa} x_i^\kappa = 1, \quad \kappa = 1, \ldots, K, \\
& \quad \sum_{i \in I_{\kappa_1}^+} x_i^\kappa_1 + \sum_{i \in I_{\kappa_2}^-} x_i^\kappa_2 \leq 1, \quad \kappa_1, \kappa_2 \in \{1, \ldots, K\} \text{ and } j \text{ an arc of } G,
\end{align*}
\]

where \( I_{\kappa_1}^+ \) represents the indices of chains \( p^\kappa_i \) such that arc \( j \) of \( G \) is used in \( p^\kappa_i \) in a positive (or negative for \( j^- \)) direction. Constraints (1c) ensure that in a feasible solution there is exactly one chain for every pair of nodes. Constraints (1d) express the condition that an arc cannot belong to two chains utilizing the arc in opposite directions. An additional condition is frequently imposed so that the entire graph, including the arcs not in any optimum chain, can be oriented so as to be strongly connected. This condition is assured by adjoining to the set of source-sink pairs the reverse \((d_\kappa, s_\kappa)\) of \((s_\kappa, d_\kappa)\) if it is not already present. In this way, the directions given by the arcs \( x_i^\kappa = 1 \) can be extended to a strongly connected graph. In step 6 below, the specified change in orientation can then always be done so that the directed graph obtained is strongly connected. Observe that if most arcs are one-way links, problem (1) may be quite hard to solve because of the enormous number of constraints and variables which may be present. The method described later is well-adapted to this type of problem because the linear programs to be solved, whose number is finite, have a smaller number of variables and constraints. Let \( \lambda_{\kappa}, \kappa = 1, \ldots, K, \) denote the dual variables for constraints (1c) and let \( \lambda_{K+}, \) denote the dual variable for the \( r \)-th constraint of (1d).

The steps of the procedure can be summarized in the following way. Let \( t \) denote the iteration number in each LP solution procedure.

**Step 1** (First feasible solution). Set \( t = 0 \). Find the minimum cost paths \( p^\kappa_i \) for each pair of nodes, on a given directed graph. Denote by \( P_t \) the set of such paths. Problem (1) without constraints (1b) and (1d) is solved by letting the variables \( x_i^\kappa = 1 \). Let \( \lambda_i^t = c_i^\kappa \) and go to step 2.

**Step 2** (Column-generation). Let \( t = t + 1 \). For each pair of nodes, find the minimum cost chain on the undirected graph; let us suppose it is \( p^\kappa_i \). Select an index \( \kappa_0 \) such that \( \Delta_t = \lambda_{\kappa_0} - c_i^\kappa_0 = \max_{\kappa} \{ \lambda_\kappa - c_i^\kappa \} \). If \( \Delta_t = 0 \) the procedure stops because an optimal solution has been reached. Otherwise adjoin the new chain \( p^\kappa_i \) to the set \( P_{t-1} \) and update the set \( P_t = P_{t-1} \cup \{ p^\kappa_i \} \). Check whether a conflict situation happens between the \( r \)-th and \( s \)-th pairs of nodes, for some \( r \) and \( s \). If this happens go to step 3; if not go to step 6.