An $O(n \log n)$-Algorithm for Solving a Special Class of Linear Programs*

W. Bein, Albuquerque, P. Brucker, Osnabrück

Received June 7, 1988; revised February 13, 1989

Abstract — Zusammenfassung

An $O(n \log n)$-Algorithm for Solving a Special Class of Linear Programs. An $O(n m + n \log n)$-algorithm was developed by Kern [1986] to solve linear programs of the form $\max \{ c x \mid l \leq A x \leq b, L \leq x \leq U \}$ where $l, b, L, U$ are nonnegative and $A$ is a 0-1-matrix of dimension $m \times n$ with the property that the support of each row is contained in the support of every subsequent row. We will show that a more general class of linear programs can be solved in $O(n \log n)$-time.

AMS Subject Classifications: 90C05, 90C35.
Key words: Linear programming, network flows, tree.

Ein $O(n \log n)$-Algorithmus zur Lösung einer speziellen Klasse von linearen Programmen. Kern [1986] entwickelte einen $O(n m + n \log n)$-Algorithmus zur Lösung von linearen Programmen der Form $\max \{ c x \mid l \leq A x \leq b, L \leq x \leq U \}$, wobei $l, b, L, U$ nichtnegativ sind und $A$ eine 0-1-Matrix der Dimension $m \times n$ mit der Eigenschaft, daß der Träger jeder Zeile von $A$ im Träger jeder der nachfolgenden Zeilen enthalten ist, darstellt. Wir zeigen, daß eine allgemeinere Klasse von linearen Programmen sich mit einem Aufwand von $O(n \log n)$ lösen läßt.

1. Introduction

Let $A = (a_{ij})$ be a binary matrix with $m$ rows and $n$ columns. For each $i = 1, \ldots, m$ denote by $\text{sup}(i)$ the support of row $i$, i.e.

$$\text{sup}(i) = \{ j \mid a_{ij} = 1 \}.$$ 

$A$ has the "Manhattan Skyline" property (MSP) if $\text{sup}(i) \subseteq \text{sup}(j)$ for all $i, j$ with $i \leq j$. Kern [1986] developed an $O(n m + n \log n)$-algorithm for linear programs of the form

$$\max cx \quad \text{subject to} \quad \begin{cases} l \leq A x \leq b \\ L \leq x \leq U \end{cases}$$

where $A$ is a binary $n \times m$-matrix with the MSP. A similar result can be found in Erenguc [1986].

* Supported in part by the Deutsche Forschungsgemeinschaft (Project COdiS) and by UCR grant 452-2810.
We will present an $O(n \log n)$-algorithm for the larger class of problems (1) in which $A$ satisfies the property

$$\sup(i) \cap \sup(j) \neq \emptyset \text{ implies } \sup(i) \subseteq \sup(j) \text{ or } \sup(j) \subseteq \sup(i). \quad (2)$$

We incorporate the $n$ restrictions $L \leq x \leq U$ into the system $l \leq Ax \leq b$ and add a restriction

$$\sum_{i=1}^{n} l_i \leq \sum_{i=1}^{n} x_i \leq \sum_{i=1}^{n} b_i$$

if $A$ does not contain a row $1 = (1, \ldots, 1)$. We also assume that all rows in the extended matrix are different. Then we get a problem of the general form

$$\max c^T x \quad \text{subject to} \quad l \leq Ax \leq b \quad (3)$$

where $A$ satisfies (2) and contains all unit vectors as rows as well as the row $1 = (1, \ldots, 1)$. In addition to this all rows are different.

Problem (3) corresponds to a network flow problem in a tree. The rows of $A$ correspond to the vertices of the tree. Vertex $i$ is a son of vertex $j$ if and only if $\sup(i) \subseteq \sup(j)$ and there exists no $k \neq i, j$ such that $\sup(i) \subseteq \sup(k) \subseteq \sup(j)$. Rows $i$ with $|\sup(i)| = 1$ correspond with the leaves of the tree. Row 1 corresponds with the root. All arcs are directed towards the root (i.e. we have an intree). For each node $i$ there is a lower capacity $l_i$ and an upper capacity $b_i$ for the flow passing $i$. We have to send flows $x_i$ from the leaves $i$ to the root of the tree such that the sum $\sum c_i x_i$ is maximized.

In Brucker [1984] it is shown that this network flow problem can be solved in $O(n \log n)$ steps if $l = 0$. We will extend this result by developing an $O(n \log n)$-algorithm for the general problem.

2. Flows in Treelike Networks with Lower Capacities

Let $T$ be an intree with nodes $1, \ldots, n$. Associated with the nodes there are lower and upper capacities $l_i$ and $b_i$ with $0 \leq l_i \leq b_i$. We assume that the nodes of $T$ are enumerated topologically. Thus, $n$ is the root of $T$. Furthermore $L(i)$ is used to denote the set of leaves of the subtree rooted at node $i (i = 1, \ldots, n)$. For all leaves $i \in L(n)$ we have “profit”-values $c_i$.

We consider the following flow problem with lower bounds

$$\max \sum_{i \in L(n)} c_i y_i \quad \text{subject to} \quad l_i \leq y_i := \sum_{j \in L(i)} y_j \leq b_i \quad i = 1, \ldots, n. \quad (4)$$

To solve (4) we denote the set of all predecessors of node $i \notin L(n)$ by $P(i)$ and assume that

$$\sum_{j \in P(i)} l_j \leq l_i \quad \text{for all } i \notin L(n). \quad (5)$$

This can always be accomplished in linear time by going up the tree and increasing lower capacities of fathers if necessary.