NOTICE CONCERNING COPYRIGHT RESTRICTIONS

The copyright law of the United States [Title 17, United States Code] governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the reproduction is not to be used for any purpose other than private study, scholarship, or research. If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that use may be liable for copyright infringement.

The institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law. No further reproduction and distribution of this copy is permitted by transmission or any other means.
Borrower: EYM
Lending String: *GZM,GZM,OSU,OSU,UIU
Patron: Pettie, Seth
Journal Title: Soviet mathematics - Doklady.
Volume: 10 Issue: 
Month/Year: 1969 Pages: 1324--1326
Article Author:
Article Title: Dinic, E., Kronrod, M. A.; An algorithm for solving the assignment problem
OCLC Number: 5417745
ILL # / Wiscat # - 82932399

Location: Math
Call #: LH 7S073
Request Date: 20111012
MaxCost: $26.00IFM
Shipping Address:
University of Michigan
106 Hatcher Graduate Library - ILL
913 South University Ave.
Ann Arbor, MI 48109-1205
Fax: (734) 936-3630
Ariel: 141.211.175.21
PDF: interlibrary.loan@umich.edu
Borrowing Notes:
Copyright Compliance: CCG

ODYSSEY

Wisconsin Library Services (WiLS)

Please report all Document Delivery Problems within 48 hours of receipt

Return this sheet via ARIEL: 144.92.126.152 or EMAIL: gzmlend@wils.wisc.edu

Briefly state the problem with the document:

This material may be protected by copyright law (Title 17 U.S. Code).
AN ALGORITHM FOR THE SOLUTION OF THE ASSIGNMENT PROBLEM

UDC 518.5

E. A. DINIC AND M. A. KRONROD

Let \((a_{ij})\) be a square matrix of order \(n\). A solution of the assignment problem for this matrix is a set of \(n\) elements of the matrix, one in each row and each column, such that the sum of these elements is minimal with respect to all such sets. Algorithms are known which solve the assignment problem after \(Cn^2\) operations, for example the Bradford method.

In the present article an algorithm is constructed which solves the problem more rapidly: one modification after \(Cn\log n\) operations, the other after \(Cn^2\) operations.

Definition. Let some vector \(\Delta = (\Delta_1, \Delta_2, \cdots , \Delta_n)\) be given. An element \(a_{ij}\) is called \(\Delta\)-minimal, if \(a_{ij} < \Delta_i - \Delta_j\) for all \(k\).

Lemma. For any \(\Delta\) let there be given a set of \(n\) minimal elements \(a_{1(1)}, a_{2(2)}, \cdots , a_{i(i)}\), one from each row and column. Then this set is a solution of the assignment problem.

Proof.\[\sum_{i=1}^{n} a_{i(i)} = \sum_{i=1}^{n} \Delta_i + \sum_{i=1}^{n} (a_{i(i)} - \Delta_i) = 0.\]

The first sum in the right member is constant for all sets, and the second is minimal.

For any vector \(\Delta\) let us select in each row one of the \(\Delta\)-minimal elements (we shall call these selected elements basics). The number of free columns (columns without basics) is called the deficiency of the set of basics. From the lemma it follows that the set of basics with deficiency zero is a solution of the problem.

An algorithm is derived below permitting iteration: for a given vector \(\Delta\) and a set of basics with nonzero deficiency, find a new vector and a set of basics possessing a lesser deficiency.

Taking the zero vector as \(\Delta\) and some set of basics for it, we can solve the problem by successive iteration.

Iteration. For any vector \(\Delta\) let us have a set of basics \(a_{1(1)}, a_{2(2)}, \cdots , a_{i(i)}\), with deficiency \(n \neq 0\). We single out some free column. Let its index be \(s\). We increase \(\Delta_s\) to the maximum \(\delta\) such that all basics remain \(\Delta\)-minimal elements (a method of finding \(\delta\) is indicated below). We obtain that for some \(i\), \(a_{i(i)} - \Delta_{i(s)} = a_{s(s)} - \Delta_s\). I.e. the element \(a_{i(s)}\) remains \(\Delta\)-minimal. We call it an alternative basic and single it out a column with index \(s_{\delta} = \delta/s\), containing a basic of this row (we now have two marked columns). We increase \(\Delta_{s_{\delta}}\) and \(\Delta_s\) to the maximal \(\delta\) such that all basics remain \(\Delta\)-minimal, we find a new alternative basic in one of the columns and single out a new column, and so on until we single out a column with two or more basics.

Now we shall construct a new set of basics. We observe that in each row there is not more than one alternative basic.

We shall call a change of basic in a row the following operation: an alternative basic in this row is declared basic, and the old basic ceases to be basic. We shall produce a change of basic in a row in which the last alternative basic lies. In this connection, in the last singled out column the number of basics is decreased by 1. In the column where the new basic appears we take the old basic and effect a change of basics in the corresponding row, and so on until a basic appears in the column with index \(s_{\delta}\). Thereby we have obtained the vector \(\Delta\) we require and the set of basics for it with deficiency equal to \(n-1\).

It is easy to see that each iteration of the described algorithm requires \(Cn^2\) operations, not counting operations entering into the calculation of \(\delta\) at each stage of the iteration. Inasmuch as the number of iterations does not exceed \(n\), only the time of computation of \(\delta\) is of interest to us.

We shall now indicate various methods of computing \(\delta\). Let us consider an arbitrary stage of some iteration.

Let \(S\) be the set of indices of the marked columns, and \(R\) the set of indices of those rows in which there are no alternative basics. Then we have \(\delta = \min_{i \in S, j \in R} (a_{i(j)} - \Delta_j - (a_{i(j)} - \Delta_i))\). If at each step of the iteration to compute \(\delta\) directly by this formula as the minimum of a group of a length of order \(n^2\), then in all the algorithm requires \(Cn^3\) operations.

We can write \(\delta = \min_{i \in S} \min_{j \in R} (a_{i(j)} - \Delta_j - (a_{i(j)} - \Delta_i))\); this formula is used for the first modification of the algorithm which permits one rapidly to find \(\min_{i \in S} \min_{j \in R} (a_{i(j)} - \Delta_j - (a_{i(j)} - \Delta_i))\) and which requires \(Cn^2 \log n\) operations, or to write \(\delta = \min_{i \in S} \min_{j \in R} (a_{i(j)} - \Delta_j - (a_{i(j)} - \Delta_i))\); this formula is used for the second modification, which permits one quickly to compute \(\min_{i \in S} \min_{j \in R} (a_{i(j)} - \Delta_i)\) and requires \(Cn^3\) operations.

First modification. We introduce a certain method of listing the ordering of the group of numbers without permutation of its terms, permitting quick reaction to cancellation (without re-enumeration) of its elements.

Let a group of \(n\) numbers be given. For listing of the ordering, a group of \(n\) boxes of information is required. In the box corresponding to some element of the group we place the indices to the preceding and succeeding (in the ordering sense) elements, i.e., their numbers.

Let it be required for us to delete a certain element of the group. From its information box we know the indices of its neighbors in the ordering. We change their information boxes, placing instead of the indicators to the deleted element the indicators to its opposite neighbor. It is obvious that the remaining group will be a list of its ordering in the sense indicated above.

For the initial ordering \(Cn \log n\) operations are required, for the change associated with a deletion \(Cn\) operations are required.

We use this method for rapid finding of \(\min_{i \in S} \min_{j \in R} (a_{i(j)} - \Delta_j - (a_{i(j)} - \Delta_i))\) for each \(k \in S\).

Let \(b_{k}\) be \(\min_{i \in S} \min_{j \in R} (a_{i(j)} - \Delta_j - (a_{i(j)} - \Delta_i))\). Each of the columns of \(b_{k}\) we order and list this ordering according to columns of a table of \(n \times n\) information boxes. We write out separately the row of minima by columns, equal to \(b_{k}\) at the beginning of the iteration.

In the process of execution of an iteration we must delete from the matrix \(\delta\) the rows in which alternative basics appear, in order that the minimum by columns of the remaining matrix will be the sought \(\min_{i \in S} \min_{j \in R} \delta_{ij}\). The corresponding change in the information table and the row of minima, and also the finding of \(\delta\) according to the formula \(\delta = \min_{i \in S} \min_{j \in R} b_{ij}\) requires \(Cn\) operations at each step of the iteration. But because the composition of the information table for each iteration requires \(Cn^3 \log n\) operations, the total course of the algorithm requires \(Cn^3 \log n\) operations.
Second modification. We have \( \vartheta = \min_{x \in [0,1]} \{ \min_{y \in [0,1]} (\alpha_{yk} - 2y - (\alpha_{yk} - 2y)) \} \). Let us consider the vector \( q = (q_1, q_2, \ldots, q_n) \), where \( q_i = \min_{x \in [0,1]} (\alpha_{ik} - 2x) \). We have \( \vartheta = \min_{x \in [0,1]} (\alpha_{ik} - 2x) \).

From this formula it follows that for solution of the problem after \( Cn^2 \) operations it is sufficient to be able to calculate at each step the vector \( q \) after \( Cn^2 \) operations. We shall show how to do this.

At the first step of the iteration the vector \( q \) coincides with the first column marked. At each succeeding step the vector \( q \) is obtained from the old vector \( q \) also from a new column marked according to the following formula: \( q' = \min (a_{nk}; 2x) \), where \( s_n \) is the index of the marked column.

Moscow State University

Received 27/FEB/69

Translated by:
R. F. Rinchard

Dokl. Akad. Nauk SSSR
Tom 189 (1969), No. 1

Vol. 10 (1969), No. 6

ON THE QUASI-EQUIVALENCE OF BASES IN THE INFINITE CENTER
OF THE HILBERT SCALE

UDC 513.88
V. P. ZAMARUTA

1°. Let \( x_1, x_2, \ldots \) be an unconditional Schauder basis in a linear topological space \( E \). Clearly, the system

\[ y_n = \lambda_n T(x_n) \]

is an unconditional basis in \( E \) for any permutation of the natural numbers \( \{x_n\} \), sequence of scalars \( \lambda_n \neq 0 \), and isomorphism \( T : E \to E \). The bases \( \{x_n\} \) and \( \{y_n\} \) are called quasi-equivalent [1, 2].

The natural question arises: are all unconditional bases quasi-equivalent in \( E \)? This problem was answered affirmatively for the entire class of nuclear countably normed spaces \( [1,3] \).

The assumption of the necessity of the space was removed in [5] for the case of the so-called finite centers of Hilbert scales.** In this paper we establish the quasi-equivalence of all unconditional bases in the infinite centers of completely continuous Hilbert scales. The proof of the quasi-equivalence in infinite centers differs qualitatively from the arguments of [5]. This is related to the fact that Lemma 3 of [5] has no analog for infinite centers.

2°. Every positive selfadjoint bounded operator \( A \) on a Hilbert space \( H \) generates a Hilbert scale \( \{H_n\} \) of Hilbert spaces \( H_n \) with scalar products

\[(x,y)_{H_n} = (A^{-n}x, A^{-n}y),\]

where \((x,y)\) is the scalar product in \( H = H_0 \). A scale \( \{H_n\} \) is called completely continuous if the operator \( A \) is completely continuous. In this case we can assume that \( H_n = \|x\|_{H_n} \), where \( \|x\|_{H_n} = \sqrt{x^*x} \), where \( x^* = \lambda_n\), the \( \lambda_n \) are the proper (eigen) values of \( A \) decreasingly ordered, i.e. \( \lambda_n \leq \lambda_{n+1} \).

Following [1], the countable Hilbert space \( E = \lim_{n \to \infty} (x^n, y^n) \) is called the center of the scale \( \{H_n\} \). Finite \( H = \omega \) and infinite if \( \omega = \infty \). For \( x = (x_1, \ldots, x_n, \ldots) \in H_\omega \) we will write

\[ \|x\|_{H_\omega} = \sup_{n} \left( \sum_{k=1}^{n} \|x_k\|^2 \right)^{1/2}. \]

If \( H_n \supset H_1 \) are two Hilbert spaces with (completely) continuous embedding, then there exists a (completely continuous) Hilbert scale \( \{G_n\} \) such that \( G_n \supset H_n \). \( G_1 = H_1 \). We will say that \( \{G_n\} \) is the scale stretched over \( H_n \) and \( H_1 \) and use the notation \( G_n = (H_n)^{\lambda} \).

We will repeatedly use the following simple consequence of a known interpolation theorem [7],

** This question was first raised by M. G. Haimov for a space of analytic functions.

Lemma (cf. [9]), Let \( H \supset H_1 \supset H_2 \supset H_3 \) be four Hilbert spaces with (completely) continuous embeddings. Then for any \( 0 < p < 1 \), we have the (completely) continuous embedding:

\[ (H_1)^{\lambda} \supset (H_2)^{\lambda} \supset (H_3)^{\lambda}. \]