ELSEVIER

Discrete Applied Mathematics 52 (1994) 71–82

DISCRETE APPLIED MATHEMATICS

Recognition of d-dimensional Monge arrays

Rüdiger Rudolf[1]

Institut für Mathematik B, Technical University Graz, Kopernikusgasse 24, A-8010 Graz, Austria

(Received 18 September 1992)

Abstract

It is known that the d-dimensional axial transportation (assignment) problem can easily be solved by a greedy algorithm if and only if the underlying cost array fulfills the d-dimensional Monge property. In this paper the following question is solved: Is it possible to find d permutations in such a way that the permuted array becomes a Monge array? Furthermore we give an algorithm which constructs such permutations in the affirmative case. If the cost array has the dimensions $n_1 \times n_2 \times \cdots \times n_d$ with $n_1 \leqslant n_2 \leqslant \cdots \leqslant n_d$, then the algorithm has time complexity $O(d^2 n_2 \cdots n_d(n_1 + \log n_d))$. By using this algorithm a wider class of d-dimensional axial transportation problems and in particular of the d-dimensional axial assignment problems can be solved efficiently.

Key words: Monge arrays; Transportation problems

1. Introduction

Already in the 18th century Monge [12] observed that if unit quantities are to be transported from locations X and Y to Z and W in such a way as to minimize the total distance traveled, then the route from X and the route from Y must not intersect. This property was rediscovered by Hoffman in 1963 [11]. Hoffman calls an $n \times m$ matrix C a Monge matrix if C satisfies

$$c_{ij} + c_{rs} \leqslant c_{is} + c_{rj} \quad \forall i < r,\ \forall j < s. \qquad (1)$$

Property (1) is commonly known as the Monge property—named after Monge—although in some papers, e.g. [6, 7], property (1) is called strong Monge property to distinguish it from a weaker condition defined for square matrices. Another condition

[1]The author acknowledges financial support by the Fonds zur Förderung der wissenschaftlichen Forschung, Project P8971-PHY.

SSDI 0166-218X(92)00189-A

related to (1) leads to the definition of Monge sequences, which are treated e.g. in [3, 10, 11].

Aggarwal and Park [1, 2] generalize property (1) to d-dimensional arrays:

Definition 1. For $d \geqslant 2$, an $n_1 \times n_2 \times \cdots \times n_d$ d-dimensional array $C = (c_{i_1 i_2 \ldots i_d})$ has the *Monge property* if for each pair of entries $c_{i_1 i_2 \ldots i_d}$ and $c_{j_1 j_2 \ldots j_d}$ we have

$$c_{s_1 s_2 \ldots s_d} + c_{t_1 t_2 \ldots t_d} \leqslant c_{i_1 i_2 \ldots i_d} + c_{j_1 j_2 \ldots j_d}, \tag{2}$$

where for $1 \leqslant k \leqslant d$, $s_k := \min\{i_k, j_k\}$ and $t_k := \max\{i_k, j_k\}$.

Arrays possessing property (2) are called Monge arrays. Monge arrays play an important role as cost arrays in connection with transportation problems. Hoffman [11] shows that the lexicographical greedy algorithm — in this case nothing other than the well known north-west corner rule [9] — solves the classical two-dimensional transportation problem if and only if the cost matrix is a Monge matrix.

The classical transportation problem can easily be extended to d dimensions. The d-dimensional transportation problem (dTP) is defined as follows: Given an $n_1 \times n_2 \times \cdots \times n_d$ array C, d supply–demand vectors $A_1, \ldots, A_d$, where the kth vector $A_k = \{a_k(i)\}$ contains n_k entries. All elements of A_k are positive and

$$\sum_{i=1}^{n_1} a_1(i) = \sum_{i=1}^{n_2} a_2(i) = \cdots = \sum_{i=1}^{n_d} a_d(i).$$

Then the problem is to

$$\text{minimize} \quad \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_d=1}^{n_d} c_{i_1 i_2 \ldots i_d} \cdot x_{i_1 i_2 \ldots i_d}$$

$$\text{such that} \quad \sum_{\substack{i_1, i_2, \ldots, i_d \\ i_k = t}} x_{i_1 i_2 \ldots i_d} = a_k(t), \quad 1 \leqslant k \leqslant d, \ 1 \leqslant t \leqslant n_k,$$

$$\text{where} \quad x_{i_1 i_2 \ldots i_d} \geqslant 0 \quad \forall i_1, i_2, \ldots, i_d.$$

We get the formulation of the closely related d-dimensional axial assignment problem, (dAP), if $n_1 = n_2 = \cdots = n_d$, $a_k(i) = 1$ and all $x_{i_1 i_2 \ldots i_d}$ are forced to be integer.

Bein et al. [4] prove a theorem, which can be seen as the natural generalization of Hoffman's result. They show that the lexicographical greedy algorithm solves the d-dimensional transportation problem if and only if C is a Monge array. Therefore (dTP) can be solved in $O(\sum_{k=1}^{d} n_k)$ time if C fulfills (2). Though (dAP) is in general NP-hard, it can even be solved in $O(n)$ time.

In this paper a wider class of efficiently solvable (dTP)'s and polynomially solvable (dAP)'s is presented based on the concept of Monge arrays. We ask whether an array can be permuted into a Monge array just by renumbering the occurring variables in a different order. If such d permutations can be found, the problem can be solved by the lexicographical greedy algorithm with respect to the accordingly permuted array.

A method for determining two permutations ϕ and ψ for an arbitrary $n \times n$ matrix is due to Deineko and Filonenko [8] and has time complexity $O(n^2)$. In [7], Cechlárová and Szabó outline a recognition algorithm for square matrices with respect to a different weaker Monge property.

In this paper we search all d-tuples $(\phi_1, \phi_2, \ldots, \phi_d)$ of permutations, such that the permuted array $C_{\phi_1,\phi_2,\ldots,\phi_d}$, defined as $C_{\phi_1,\phi_2,\ldots,\phi_d} := (c_{\phi_1(i),\phi_2(i),\ldots,\phi_d(i)})$ becomes a Monge array. An algorithm is presented, which either constructs all d-tuples $(\phi_1, \phi_2, \ldots, \phi_d)$ such that $C_{\phi_1,\phi_2,\ldots,\phi_d}$ is Monge or proves that no such permutations can exist.

The paper is organized as follows: Section 2 outlines the algorithm of Deineko and Filonenko which constructs for a given matrix C one pair (ϕ, ψ) such that $C_{\phi,\psi}$ fulfills the Monge property. In Section 3 we first investigate the set of all permutations (ϕ, ψ) such that a given $n \times m$ matrix becomes Monge, when ϕ is applied to the rows and ψ to the columns of C. Based on these results we then describe the algorithm for d dimensions. To illustrate it two short examples are given in Section 4.

2. The algorithm for two dimensions

Since property (1) can also be written as

$$c_{ij} + c_{i+1,j+1} \leqslant c_{i,j+1} + c_{i+1,j}, \quad 1 \leqslant i \leqslant n-1, \ 1 \leqslant j \leqslant m-1 \tag{3}$$

an $n \times m$ matrix C can be checked in $O(nm)$ time to be a Monge matrix or not.

A method for determining permutations ϕ and ψ for an arbitrary $n \times n$ matrix—which can simply be extended to $n \times m$ matrices—is due to Deineko and Filonenko [8]. Their algorithm has time complexity $O(n^2)$ and runs as follows.

Algorithm 2.1. Determination of (ϕ, ψ) such that $C_{\phi,\psi}$ is Monge.

Input: an $n \times n$ matrix C.

Output: a pair (ϕ, ψ) s.t. $C_{\phi,\psi}$ is a Monge matrix.

(1) Find an integer j such that

$$c_{1k} - c_{jk} \neq c_{1l} - c_{jl} \quad \text{for some } k, l.$$

If there is no such j, then C is a "constant" matrix, i.e. $c_{ij} := u_i + v_j$ with $u_i := c_{i1}$, $v_j := c_{1j} - c_{11}$ and ϕ and ψ can be chosen arbitrarily.

(2) Determine $\tilde{\psi}$ by sorting $(c_{1i} - c_{ji})$:

$$c_{1\tilde{\psi}(1)} - c_{j\tilde{\psi}(1)} \leqslant c_{1\tilde{\psi}(2)} - c_{j\tilde{\psi}(2)} \leqslant \cdots \leqslant c_{1\tilde{\psi}(n)} - c_{j\tilde{\psi}(n)}.$$

(3) Find the maximal value of m such that

$$c_{1\tilde{\psi}(l)} - c_{j\tilde{\psi}(l)} = \min_{k=1,\ldots,n} (c_{1k} - c_{jk}) \quad \forall l = 1, \ldots, m$$

and the minimal value of M such that

$$c_{1\tilde{\psi}(L)} - c_{j\tilde{\psi}(L)} = \max_{k=1,\dots,n} (c_{1k} - c_{jk}) \quad \forall L = M, \dots, n.$$

(4) Construct an auxiliary vector $R = \{r_i\}$ with elements

$$r_i := (n - M + 1) \sum_{l=1}^{m} c_{i\tilde{\psi}(l)} - m \sum_{L=M}^{n} c_{i\tilde{\psi}(L)} \quad \forall i = 1, \dots, n.$$

(5) Determine ϕ by sorting r_i with

$$r_{\phi(1)} \leqslant r_{\phi(2)} \leqslant \cdots \leqslant r_{\phi(n)}$$

and find ψ by sorting $(c_{\phi(1)i} - c_{\phi(n)i})$:

$$c_{\phi(1)\psi(1)} - c_{\phi(n)\psi(1)} \leqslant c_{\phi(1)\psi(2)} - c_{\phi(n)\psi(2)} \leqslant \cdots \leqslant c_{\phi(1)\psi(n)} - c_{\phi(n)\psi(n)}.$$

(6) If and only if the $(n-1)^2$ inequalities for $C_{\phi,\psi}$ given by (3) are satisfied then $C_{\phi,\psi}$ is a Monge matrix.

If we use an extended version of above algorithm to arbitrary $n \times m$ matrices, we get an overall time complexity of $O(nm + n \log n)$, if $m \leqslant n$.

3. The algorithm for three and higher dimensions

The idea of the algorithm for permuting a d-dimensional array to become a Monge array is on one hand based on the algorithm of Deineko and Filonenko and on the other hand on the following result, first proved by Aggarwal and Park.

Lemma 3.1 (Aggarwal and Park [1, 2]). *A d-dimensional array C is a Monge array if and only if every two-dimensional submatrix is a Monge matrix.*

Let us start with the main idea for the algorithms for d dimensions. First we apply Algorithm 2.1 to each two-dimensional $n_i \times n_j$ submatrix of the given $n_1 \times n_2 \times \cdots \times n_d$ array C, to get a pair of permutations which transforms this submatrix into a Monge matrix.

If at least one submatrix cannot be rearranged as a Monge matrix, then C can never be a Monge array (refer to Lemma 3.1). Now we have to check if there is a d-tuple of permutations $(\phi_1, \phi_2, \dots, \phi_d)$ such that $C_{\phi_1,\phi_2,\dots,\phi_d}$ fulfills property (2).

For this purpose it is not sufficient to know for every $n_i \times n_j$ submatrix of the array C the pair of permutations (ϕ_i, ϕ_j) determined by Algorithm 2.1, we need the set of all permutations which transform this submatrix into a Monge matrix. Therefore, we shall describe in the following the set of all permutations P_C defined as $P_C := \{(\phi, \psi) \mid C_{\phi,\psi} \text{ is a Monge matrix}\}$ for a given $n \times m$ matrix C. Fortunately, it turns out that the set P_C can be characterized in a nice way.

First we need two definitions.

Definition 3.2. Let ϕ be a permutation of $\{1,\dots,n\}$. Then ϕ^- defined as

$$\phi^-(i) := \phi(n-i+1) \quad \forall i = 1,\dots,n$$

is called the *reverse* permutation of ϕ.

Definition 3.3. Given an $n \times m$ matrix C. Then two rows i and j are called *equivalent* if there exists a constant number a such that $c_{il} + a = c_{jl}$ $\forall l = 1,\dots,m$. Two columns i and j are called *equivalent* if there exists a constant number b such that $c_{li} + b = c_{lj}$ $\forall l = 1,\dots,n$. 兩列/行差固定，則稱為等價

Now we are ready to formulate the following observations about P_C.

Observation 3.4. *Let an $n \times m$ matrix C be given. Then*

$$(\phi,\psi) \in P_C \Leftrightarrow (\phi^-,\psi^-) \in P_C.$$ 順<->逆

Observation 3.5. *Let C be an $n \times m$ Monge matrix. Then two rows (columns) i and j can change their position in C not losing the Monge property if and only if rows (columns) i and j are equivalent.*

對調等價

Proof. ($\Rightarrow$) Let $r < s$. Since C is a Monge matrix the inequality $c_{ir} + c_{js} \leqslant c_{is} + c_{jr}$ holds. Exchanging row i and row j not losing property (1) means that $c_{jr} + c_{is} \leqslant c_{js} + c_{ir}$. But this implies that $c_{jr} + c_{is} = c_{js} + c_{ir} \Leftrightarrow c_{jr} - c_{js} = c_{ir} - c_{is}$. Since r and s were chosen arbitrarily, rows i and j are equivalent.

($\Leftarrow$) Since rows i and j are equivalent, there exists a constant number a such that $c_{ir} + a = c_{jr}$ $\forall r$. Let $r < s$. Then again $c_{ir} + c_{js} \leqslant c_{is} + c_{jr}$ holds. Adding on both sides $2a$ yields $c_{jr} + c_{is} \leqslant c_{js} + c_{ir}$. In a similar way all other inequalities can be verified to ensure property (1). Thus our observation is proven. □

Observation 3.6. *Let an $n \times m$ Monge matrix C be given. Let $1 \leqslant i < j < k \leqslant n$ and assume that row i and row k are equivalent. Then row j is equivalent to row i and row k.*

夾擠等價

Proof. Let $r < s$. Then the following inequalities hold:

$$c_{ir} + c_{js} \leqslant c_{is} + c_{jr} \quad \text{and} \quad c_{jr} + c_{ks} \leqslant c_{js} + c_{kr}.$$

Adding these two we obtain: $c_{ir} + c_{ks} \leqslant c_{is} + c_{kr}$. On the other hand, since row i and row k are equivalent, $c_{ir} + c_{ks} = c_{is} + c_{kr}$. Hence we have $c_{ir} + c_{js} = c_{is} + c_{jr} = c_{is} + c_{kr}$. Since r and s were arbitrary, all three rows are equivalent. □

From Observation 3.6 it follows that in a Monge matrix equivalent rows and columns occur consecutively in a block of the matrix and inside this block they can have an arbitrary order (refer to Observation 3.5). Therefore we can restrict our further investigations on matrices without equivalent rows and columns.

對於沒有等效行和列的 n x m 矩陣，矩陣的反轉是在不破壞 Monge 結構的情況下對其進行反轉的唯一方法。

What we want to prove is that for an $n \times m$ matrix without equivalent rows and columns the reversing of the matrix is the only way to permute it without destroying the Monge structure. To this end we need the following two lemmata.

Lemma 3.7. *Let C be a 2×3 Monge matrix without equivalent rows and equivalent columns and denote I_n the identity permutation on $\{1, \ldots, n\}$. Then*

$$P_C = \{(I_2, I_3), (I_2^-, I_3^-)\}.$$

Proof. Since C is a Monge matrix and since Observation 3.4 holds, $P_C \supseteq \{(I_2, I_3), (I_2^-, I_3^-)\}$. We only have to show that no other possible pair of permutations exists. Therefore two cases are distinguished:

(i) Row 1 and row 2 remain unchanged. But then at least two columns have to change their positions. Without loss of generality (w.l.o.g) we change column 1 and 2. To fulfill again the Monge property the inequality $c_{12} + c_{21} \leqslant c_{11} + c_{22}$ must be satisfied. On the other hand $c_{11} + c_{22} \leqslant c_{12} + c_{21}$ holds, since the original matrix is a Monge matrix. So $c_{12} + c_{21} = c_{11} + c_{22} \Leftrightarrow c_{11} - c_{12} = c_{21} - c_{22}$ implying that column 1 and column 2 are equivalent. But this is a contradiction to our assumption. Exchanging other pairs of columns leads again to contradictions.

(ii) Row 2 and row 1 change their positions. Then the only possibility to obtain a Monge matrix by permuting the columns is $\psi = \langle 3, 2, 1 \rangle$. We assume that e.g. column 1 precedes column 2 in ψ. But then we get as in (i) the equation $c_{11} + c_{22} = c_{12} + c_{21}$ which yields again a contradiction to our assumptions.

So $P_C = \{(I_2, I_3), (I_2^-, I_3^-)\}$. □

Lemma 3.8. *Let C be a 3×3 Monge matrix without equivalent rows and equivalent columns. Then*

$$P_C = \{(I_3, I_3), (I_3^-, I_3^-)\}.$$

Proof. Clearly C_{I_3, I_3} and $C_{I_3^-, I_3^-}$ are Monge matrices. It remains to show that there is no other pair of permutations $(\phi, \psi) \in P_C$. Since C contains no equivalent rows and columns and C is a Monge matrix, at least one of the following conditions is satisfied:

(i) $c_{11} + c_{22} < c_{12} + c_{21}$ and $c_{22} + c_{33} < c_{23} + c_{32}$,

(ii) $c_{12} + c_{23} < c_{13} + c_{22}$ and $c_{21} + c_{32} < c_{22} + c_{31}$.

We only treat case (i) and case (ii) can be handled in a similar way.

Since C is a Monge matrix we have $c_{21} + c_{32} \leqslant c_{31} + c_{22}$. Combining this e.g. with $c_{11} + c_{22} < c_{12} + c_{21}$ we derive $c_{11} + c_{32} < c_{12} + c_{31}$. Similarly we get the following inequalities: $c_{12} + c_{33} < c_{32} + c_{13}$, $c_{21} + c_{33} < c_{31} + c_{23}$, $c_{11} + c_{23} < c_{13} + c_{21}$. So if row 1 precedes row 2 in ϕ, $r_1 \prec r_2$ for short, then $c_1 \prec c_2$, $c_1 \prec c_3$. Again from $r_1 \prec r_3$ it follows that $c_1 \prec c_2$, $c_2 \prec c_3$ and $r_2 \prec r_3$ implies $c_1 \prec c_3$ and $c_2 \prec c_3$. It can easily be verified that the only two possible pairs of permutations are the claimed ones. For every other pair we would get a contradiction in the conditions described above. □

如果矩陣 C 已經是 Monge 矩陣，那麼在不丟失 Monge 屬性的情況下對其進行置換的唯一方法是反轉整個矩陣。

Now we are prepared to formulate a characterization of Monge matrices without equivalent rows and columns. If a matrix C is already a Monge matrix then the only way to permute it without losing the Monge property is to reverse the total matrix. More precisely we have the following result.

Theorem 3.9. *Given an arbitrary $n \times m$ Monge matrix without equivalent rows and equivalent columns. Then*

$$P_C = \{(I_n, I_m), (I_n^-, I_m^-)\}.$$

Proof. We assume that ψ is neither the identity nor its reverse permutation. But then at least one of those conditions below hold. Take a triple of integers $1 \leqslant i < j < k \leqslant m$ with

(a) $\psi(i) < \psi(j)$ and $\psi(j) > \psi(k)$ or

(b) $\psi(i) > \psi(j)$ and $\psi(j) < \psi(k)$.

We will only prove part (a), the proof of (b) can be done analogously. Let us concentrate on the $n \times 3$ matrix containing the columns $\psi(i)$, $\psi(j)$ and $\psi(k)$ in the original order before using permutation ψ. As long as a pair of equivalent rows exist delete one of them. After that, two different types of matrices can remain:

(i) A 2×3 matrix is left and since the three columns $\psi(i), \psi(j)$ and $\psi(k)$ are pairwise non-equivalent the conditions of Lemma 3.7 are satisfied. Therefore $\psi(i)$, $\psi(j)$ and $\psi(k)$ (in this order) can never be a part of a Monge matrix. Since this 2×3-matrix is a submatrix of C the whole matrix cannot be a Monge matrix.

(ii) If more than two rows are left we take the first, the last and an arbitrary row and apply Lemma 3.8. Again this submatrix cannot appear in a Monge matrix.

So no permutation ϕ exists, such that $(\phi, \psi) \in P_C$. And since I_m and I_m^- are the only permutations not fulfilling both (a) and (b) the theorem is proven. □

It turns out that instead of describing the set P_C it is easier to change over to the set $\bar{P}_C$ which is defined as follows: $\bar{P}_C := \{(\phi, \psi) | (\phi^{-1}, \psi^{-1}) \in P_C\}$.

The reason therefore is that in every member ϕ^- (ψ^-) of $\bar{P}_C$ all equivalent rows (columns) occur consecutively. Note that this property does not hold for ϕ and ψ, respectively. The key observation is that $\bar{P}_C$ can be represented in a compact form. To this end, we define a block structure. Two rows (columns) belong to the same block if and only if they are equivalent. Within a block no order is fixed, but the ordering of the blocks is fixed.

More formally we introduce the following definition.

Definition 3.10. Let a set S of permutations, a number k with $1 \leqslant k \leqslant n$ and a partition of $\{1, \ldots, n\}$ into k ordered blocks B_i, each containing b_i numbers, be given. Define $s_1 := 1, s_{i+1} := s_i + b_i$ for all $i = 1, \ldots, k$. Then S is called a *block-permutation* if the following property holds:

$$\phi \in S \Leftrightarrow \forall i = 1, \ldots, n \; \forall j = 1, \ldots, k: i \in B_j \Rightarrow s_j \leqslant \phi(i)^{-1} < s_{j+1}.$$

To illustrate Definition 3.10 consider the following block-permutation $S := \langle[1], [3, 5], [2], [4, 6]\rangle$. Each block is marked with $[\cdots]$. Then S contains those four different permutations: 1*2*1*2=4

$$\langle 1, 3, 5, 2, 4, 6\rangle, \quad \langle 1, 3, 5, 2, 6, 4\rangle, \quad \langle 1, 5, 3, 2, 4, 6\rangle, \quad \langle 1, 5, 3, 2, 6, 4\rangle.$$

Another data structure describing a similar set of permutations in a compact form are PQ-trees and can be found e.g. in [5].

Now we are prepared to formulate an algorithm for constructing the set $\bar{P}_C$.

Algorithm 3.11. Construction of the set $\bar{P}_C$.
Input: an $n \times m$ matrix C.
Output: the set $\bar{P}_C$.

(1) Use Algorithm 2.1 to determine a pair $(\phi, \psi) \in P_C$. If no such pair exists, $\bar{P}_C = \emptyset$; stop.
(2) Determine the blocks of ϕ^{-1} by scanning ϕ^{-1} element by element from left to right. Let $S(\phi)$ be the corresponding block-permutation.
(3) Determine the blocks of ψ^{-1} by scanning ψ^{-1} element by element from left to right. Let $S(\psi)$ be the corresponding block-permutation.
(4) Set

$$\bar{P}_C := \{(\sigma, \tau) \mid (\sigma \in S(\phi) \wedge \tau \in S(\psi)) \vee (\sigma^- \in S(\phi) \wedge \tau^- \in S(\psi))\},$$

which can shortly be written as

$$\bar{P}_C := (S(\phi) \times S(\psi)) \cup (S(\phi^-) \times S(\psi^-)).$$

Theorem 3.12. *Given an $n \times m$ matrix C with $n \leqslant m$. Then Algorithm 3.11 constructs $\bar{P}_C$ in $O(nm + m \log m)$ time.*

Proof. The correctness of Algorithm 3.11 follows from the previous observations and Theorem 3.9. Hence the complexity bound remains to be proven. Algorithm 2.1 can be performed in $O(nm + m \log m)$ steps. Since we have only to check adjacent rows, $S(\phi)$ can be constructed in $O(nm)$. Hence Step 3 can also be done in $O(nm)$. Step 4 needs linear time to reverse the block-permutation. Summarizing all steps we get the claimed complexity of $O(nm + m \log m)$. □

A fast algorithm for intersecting block-permutations is a first step towards an efficient algorithm for the recognition of d-dimensional Monge arrays. We first show how to intersect two arbitrary block-permutations S_1 and S_2 and how to construct the corresponding new block-permutation $S_3 := S_1 \cap S_2$. This intersection process is done recursively. Let $C_1, \ldots, C_u$ and $D_1, \ldots, D_v$ be the different blocks which define S_1 and S_2, respectively. To have $S_3 \neq \emptyset$, we must either have $C_1 \subseteq D_1$ or $D_1 \subseteq C_1$;

w.l.o.g. suppose $C_1 \subseteq D_1$. Then C_1 is the first block of the intersection S_3. Its next blocks are obtained recursively as the intersection of the block-permutations induced by $C_2, \ldots, C_u$ and $D_1 \setminus C_1, D_2, \ldots, D_v$. This algorithm can be implemented to run in linear time.

Now we show that the intersection of two pairs of block-permutations can again be represented by a new pair of block-permutations. Given two sets $\bar{P}_{C_1}$ and $\bar{P}_{C_2}$ which are represented by a pair of block-permutations for the rows and columns, say $(S(\phi_1), S(\psi_1))$ and $(S(\phi_2), S(\psi_2))$, respectively. To compute the intersection $\bar{P}$ we first determine the following intersections of pairs of block-permutations: $(S(\phi_1) \cap S(\phi_2), S(\psi_1) \cap S(\psi_2))$ and $(S(\phi_1^-) \cap S(\phi_2), S(\psi_1^-) \cap S(\psi_2))$ (note that either one set is empty or both sets are equal). Let $(S(\phi_3), S(\psi_3))$ be the resulting pair of block-permutations. Then we have $(\sigma, \tau) \in \bar{P}$ if and only if either $(\sigma, \tau) \in (S(\phi_3), S(\psi_3))$ or $(\sigma, \tau) \in (S(\phi_3^-), S(\psi_3^-))$. Hence the set $\bar{P}$ can again be represented by a pair of block-permutations and therefore constructed in linear time.

Based on Algorithm 3.11 described above and on the possibility of representing the intersection of two pairs of block-permutations again as a pair of block-permutations we are now able to give an algorithm for the main problem considered in this paper. Given an $n_1 \times n_2 \times \cdots \times n_d$ array C we want to decide whether there is a d-tuple of permutations, $(\phi_1, \phi_2, \ldots, \phi_d)$, such that $C_{\phi_1, \phi_2, \ldots, \phi_d}$ becomes a Monge array.

For the ease of exposition we first describe the algorithm for $d = 3$ and explain afterwards how it can be generalized easily to arbitrary dimension d.

Let an $n \times n \times n$ array C be given. Then Algorithm 3.13 either constructs the set of all triples of permutations ϕ, ψ and π such that $C_{\phi, \psi, \pi}$ is a Monge array or shows that no such transformation exists. It works as follows.

Algorithm 3.13. Construction of all (ϕ, ψ, π) such that $C_{\phi, \psi, \pi}$ is Monge.
Input: an $n \times n \times n$ array C.
Output: all triples (ϕ, ψ, π) s.t. $C_{\phi^{-1}, \psi^{-1}, \pi^{-1}}$ is a Monge array.

(1) Determine possible candidates for ϕ and ψ:
  (i) For all k define the matrix D_k with $d_{ij} := c_{ijk}$. Apply Algorithm 3.11 to construct the set $\bar{P}_{D_k}$.
  (ii) Set $P_{ij} := \{(\phi, \psi) \mid (\phi, \psi) \in \bar{P}_{D_k} \ \forall k\}$. Thus P_{ij} denotes the "intersection" of all block-permutations $\bar{P}_{D_k}$.

(2) Determine possible candidates for ϕ and π:
  (i) For all j define the matrix E_j with $e_{ik} := c_{ijk}$. Apply Algorithm 3.11 to construct the set $\bar{P}_{E_j}$.
  (ii) Set $P_{ik} := \{(\phi, \pi) \mid (\phi, \pi) \in \bar{P}_{E_j} \ \forall j\}$.

(3) Determine possible candidates for ψ and π:
  (i) For all i define the matrix F_i with $f_{jk} := c_{ijk}$. Apply Algorithm 3.11 to construct the set $\bar{P}_{F_i}$.
  (ii) Set $P_{jk} := \{(\psi, \pi) \mid (\psi, \pi) \in \bar{P}_{F_i} \ \forall i\}$.

(4) Construct the set $\bar{P}$ defined as

$$\bar{P} := \{(\phi, \psi, \pi) \mid (\phi, \psi) \in P_{ij}, (\phi, \pi) \in (P_{ik}, (\psi, \pi) \in P_{jk}\}.$$

If $\bar{P} = \emptyset$, the array C cannot be arranged as Monge array, otherwise for each triple $(\phi, \psi, \pi) \in \bar{P}$ the array $C_{\phi^{-1},\psi^{-1},\pi^{-1}}$ is a Monge array.

Theorem 3.14. *Algorithm* 3.13 *has time complexity* $O(n^3)$.

Proof. Step 1(i) can be performed in $O(n^3)$ since we use n times Algorithm 3.11. Since an intersection of two block-permutations can be done in $O(n)$ time and again be represented by a new block-permutation, we can intersect all n block-permutations in $O(n^2)$. Obviously Step 2 and 3 have same complexity as Step 1. Step 4 can also be executed in $O(n)$ time. Thus we get an overall time complexity of $O(n^3)$. □

In a straightforward way Algorithm 3.13 can be extended to d dimensions. The basic step is to construct the sets $P_{i_k i_l}$ represented as a pair of block-permutations, say (S_{i_k}, S_{i_l}), for all $1 \leqslant k, l \leqslant d$, $l \neq k$ and then — like in (4) — checking if a global solution can be found. This can be done for example in the following way. Represent the set $\bar{P}$ also as block-permutations, say $T_1, \ldots, T_d$, and fix $T_i = A$, where A denotes that block-permutation which contains all permutations. Now procede in a two-phase method. First choose an arbitrary pair $(S_{i_k}, S_{i_l}) \neq (A, A)$ and compute $T_k := T_k \cap S_{i_k}$ and $T_l := T_l \cap S_{i_l}$. In a second step as long as you find a pair (S_{i_k}, S_{i_l}) with either $T_k \neq A$ and $S_{i_k} \neq A$ or $T_l \neq A$ and $S_{i_l} \neq A$ compute again the new block-permutations T_k and T_l as the intersection of the old sets T_k and T_l with S_{i_k} and S_{i_l}. If no such pair can be found continue with the first phase.

After taking into account all computed pairs $(S_{i_k}, S_{i_l}) \neq (A, A)$ and intersecting them with $T_1, \ldots, T_d$ either the set $\bar{P}$ is empty if at least one intersection fails, or contains at least one d-tuple of permutations.

Since there are $\binom{d}{2} \cdot n^{d-2}$ possible two-dimensional submatrices of the given array and since Algorithm 3.11 has time complexity $O(n^2)$ we get an overall running time of $O(\binom{d}{2} \cdot n^d)$, which yields the time complexity of $O(d^2 n^d)$. If we use our algorithm on arbitrary $n_1 \times n_2 \times \cdots \times n_d$ arrays C where $n_1 \leqslant n_2 \leqslant \cdots \leqslant n_d$ holds, we get an overall running time complexity of $O(d^2 n_2 n_3 \cdots n_d (n_1 + \log n_d))$.

4. Examples

To illustrate Algorithm 3.13 we consider two different examples. In both examples we try to rearrange a given $4 \times 4 \times 3$ matrix C into a Monge matrix using Algorithm 3.13. For the ease of description we represent the array C as three 4×4 matrices.

Example 4.1.

$$C := \left(\begin{matrix} 12 & 15 & 12 & 9 \\ 16 & 17 & 16 & 15 \\ 14 & 16 & 14 & 12 \\ 16 & 17 & 16 & 15 \end{matrix}\right) \left(\begin{matrix} 6 & 12 & 6 & 0 \\ 14 & 16 & 14 & 12 \\ 10 & 14 & 10 & 6 \\ 14 & 16 & 14 & 12 \end{matrix}\right) \left(\begin{matrix} 12 & 15 & 12 & 9 \\ 16 & 17 & 16 & 15 \\ 14 & 16 & 14 & 12 \\ 16 & 17 & 16 & 15 \end{matrix}\right).$$

(1) Fixing k = 1 and performing Algorithm 2.1 on the matrix D with $d_{ij} = c_{ij1}$ we get a pair (ϕ, ψ) of permutations with

$$\phi = \langle 1, 3, 2, 4\rangle \quad \text{and} \quad \psi = \langle 2, 4, 3, 1\rangle.$$

Hence $\phi^{-1} = \langle 1, 3, 2, 4\rangle$ and $\psi^{-1} = \langle 4, 1, 3, 2\rangle$.
We construct $S(\phi) = \langle [1], [3], [2, 4]\rangle$ and $S(\psi) = \langle [4], [1, 3], [2]\rangle$. For all other k we get the same sets $S(\phi)$ and $S(\psi)$. Therefore

$$P_{ij} = (\langle [1], [3], [2, 4]\rangle \times \langle [4], [1, 3], [2]\rangle) \cup (\langle [2, 4], [3], [1]\rangle \times \langle [2], [1, 3], [4]\rangle).$$

(2) Compute $S(\phi) = \langle [1], [3], [2, 4]\rangle$ and $S(\pi) = \langle [2], [1, 3]\rangle$. Then

$$P_{ik} = (\langle [1], [3], [2, 4]\rangle \times \langle [2], [1, 3]\rangle) \cup (\langle [2, 4], [3], [1]\rangle \times \langle [1, 3], [2]\rangle).$$

(3) Determine $S(\psi) = \langle [4], [1, 3], [2]\rangle$ and $S(\pi) = \langle [2], [1, 3]\rangle$. Then

$$P_{jk} = (\langle [4], [1, 3], [2]\rangle \times \langle [2], [1, 3]\rangle) \cup (\langle [2], [1, 3], [4]\rangle \times \langle [1, 3], [2]\rangle).$$

(4) Calculate

$$P = (\langle [1], [3], [2, 4]\rangle \times \langle [4], [1, 3], [2]\rangle \times \langle [1, 3], [2]\rangle) \cup (\langle [2, 4], [3], [1]\rangle \times \langle [2], [1, 3], [4]\rangle \times \langle [1, 3], [2]\rangle).$$

So the given array C can be permuted into a Monge array. For example take $\phi = \langle 1, 3, 2, 4\rangle$, $\psi = \langle 2, 4, 3, 1\rangle$ and $\pi = \langle 1, 3, 2\rangle$.

Example 4.2.

$$C := \begin{pmatrix} 3 & 1 & 4 & 2 \\ 6 & 2 & 8 & 4 \\ 6 & 2 & 8 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix} \begin{pmatrix} 9 & 3 & 12 & 6 \\ 18 & 6 & 24 & 12 \\ 18 & 6 & 24 & 12 \\ 9 & 3 & 12 & 6 \end{pmatrix} \begin{pmatrix} 3 & 1 & 4 & 2 \\ 6 & 2 & 8 & 4 \\ 6 & 2 & 8 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}.$$

Performing Steps 1 to 3 we get the following sets:

$$P_{ij} = (\langle [1, 4], [2, 3]\rangle \times \langle [3], [1], [4], [2]\rangle) \cup (\langle [2, 3], [1, 4]\rangle \times \langle [2], [4], [1], [3]\rangle);$$

$$P_{ik} = (\langle [1, 4], [2, 3]\rangle \times \langle [2], [1, 3]\rangle) \cup (\langle [2, 3], [1, 4]\rangle \times \langle [1, 3], [2]\rangle);$$

$$P_{jk} = (\langle [3], [1], [4], [2]\rangle \times \langle [1, 3], [2]\rangle) \cup (\langle [2], [4], [1], [3]\rangle \times \langle [2], [1, 3]\rangle).$$

Intersecting these three sets, we get $P = \emptyset$. So no triple of permutations exists, which permutes C into a Monge array.

Acknowledgement

We would like to thank Vladimir Deineko for making available to us a description of his algorithm for solving our problem in two dimensions.

References

[1] A. Aggarwal and J.K. Park, Parallel searching in multidimensional monotone arrays, Research Report RC 14826, IBM T.J. Watson Research Center, Yorktown Heights, NY, August 1989. Submitted to J. Algorithms. Portions of this paper appear in Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science (1988) 497–512.

[2] A. Aggarwal and J.K. Park, Sequential searching in multidimensional monotone arrays, Research Report RC 15128, IBM T.J. Watson Research Center, Yorktown Heights, NY, November 1989. Submitted to J. Algorithms. Portions of this paper appear in Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science (1988) 497–512.

[3] N. Alon, S. Cosares, D.S. Hochbaum and R. Shamir, An algorithm for the detection and construction of Monge sequences, Linear Algebra Appl. 114/115, (1989) 669–680.

[4] W. Bein, P. Brucker, J.K. Park and P.K. Pathak, A Monge property for the d-dimensional transportation problem, Discrete Appl. Math., to appear.

[5] K.S. Booth and G.S. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-trees algorithms, J. Comput. System Sci. 13 (1976) 335–379.

[6] R.E. Burkard, Assignment problems: Recent solution methods and applications. System Modelling and Optimization, Proceedings of 12th IFTP Conference, Budapest, Hungary, 1985, Lecture Notes in Control and Information Sci., Vol. 84 (Springer, Berlin, 1986) 153–169.

[7] K. Cechlárová and P. Szabó, On the Monge property of matrices, Discrete Math. 81 (1990) 123–128.

[8] V.G. Deineko and V.L. Filonenko, On the reconstruction of specially structured matrices, Aktualnyje Problemy EVM, programmirovanije, Dnepropetrovsk, DGU, 1979 (Russian).

[9] G. Hadley, Linear Programming (Addison-Wesley, Reading, MA, 1962).

[10] B. Dietrich and R. Shamir, Characterization and algorithms for greedily solvable transportation problems, Proceedings of the First ACM–SIAM, Symposium of Discrete Algorithms, January 1990, 358–366.

[11] A.J. Hoffman, On simple linear programming problems, in: Convexity, Proceedings of Symposia in Pure Mathematics (AMS, Providence, RI, 1961) 317–327.

[12] G. Monge, Déblai et remblai, Mémoires de l'Academie des Sciences, Paris, 1781.