

A Monge Property for the d -Dimensional Transportation Problem

Wolfgang W. Bein^{*} Peter Brucker[†] James K. Park[‡] Pramod K. Pathak[§]

Abstract

In 1963, Hoffman gave necessary and sufficient conditions under which a family of $O(mn)$ -time greedy algorithms solves the classical two-dimensional transportation problem with m sources and n sinks. One member of this family, an algorithm based on the “northwest corner rule,” is of particular interest, as its running time is easily reduced to $O(m+n)$. When restricted to this algorithm, Hoffman’s result can be expressed as follows: the northwest-corner-rule greedy algorithm solves the two-dimensional transportation problem for all source and supply vectors if and only if the problem’s cost array $C = \{c[i, j]\}$ possesses what is known as the (two-dimensional) Monge property, which requires $c[i_1, j_1] + c[i_2, j_2] \leq c[i_1, j_2] + c[i_2, j_1]$ for $i_1 < i_2$ and $j_1 < j_2$. This paper generalizes this last result to a higher dimensional variant of the transportation problem. We show that the natural extension of the northwest-corner-rule greedy algorithm solves an instance of the d -dimensional transportation problem if and only if the problem’s cost array possesses a d -dimensional Monge property recently proposed by Aggarwal and Park in the context of their study of monotone arrays. We also give several new examples of cost arrays with this d -dimensional Monge property.

1 Introduction

The transportation problem (also known as the Hitchcock problem) is a classical and very general minimum-cost flow problem. In linear-programming terms, we are given an m -entry supply vector $A = \{a[i]\}$, an n -entry demand vector $B = \{b[j]\}$, and an $m \times n$ cost array $C = \{c[i, j]\}$, such that the entries of A and B are all positive and $\sum_i a[i] = \sum_j b[j]$, and we want to choose an $m \times n$ variable array $X = \{x[i, j]\}$ in order to

$$\begin{aligned} & \text{minimize} && \sum_{i,j} c[i, j] x[i, j] \\ & \text{subject to} && \sum_j x[I, j] = a[I] && \text{for } 1 \leq I \leq m \\ & && \sum_i x[i, J] = b[J] && \text{for } 1 \leq J \leq n \\ & && x[i, j] \geq 0 && \text{for } 1 \leq i \leq m \text{ and } 1 \leq j \leq n. \end{aligned}$$

(The array X is called a *solution* for the problem, and a solution X satisfying the problem’s constraints is called a *feasible solution*.)

^{*}Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, U.S.A.

[†]Fachbereich Mathematik/Informatik, Universität Osnabrück, D-4500 Osnabrück, Germany.

[‡]Algorithms and Discrete Mathematics Department (Org. 1423), Sandia National Laboratories, Albuquerque, NM 87185, U.S.A. This work was supported by the U.S. Department of Energy under Contract DE-AC04-76DP00789.

[§]Department of Mathematics and Statistics, University of New Mexico, Albuquerque, NM 87131, U.S.A.

The transportation problem has been the subject of much study over the last fifty years. One reason for this interest is the transportation problem's close relation to the (capacitated) minimum-cost flow problem. Not only is the transportation problem easily reduced to the minimum-cost flow problem, but it is also possible to reduce the minimum-cost flow problem to the transportation problem, as shown by Wagner [20].

The fastest algorithm currently known for the general transportation problem, due to Orlin [19], runs in $O(mn^2 \lg n + n^2 \lg^2 n)$ time (where we assume without loss of generality that $m \leq n$). However, under certain circumstances, the transportation problem can be solved significantly faster, in $O(mn)$ or even $O(m + n)$ time. These circumstances were first studied by Hoffman [15]; he identified necessary and sufficient conditions under which a family of $O(mn)$ -time greedy algorithms solves the transportation problem with m sources and n sinks. One member of this family, an algorithm based on the "northwest corner rule," is of particular interest, as its running time is easily reduced to $O(m + n)$. This algorithm assigns values to the variables in lexicographical order, greedily making each value as large as possible. Only a small fraction of the cost array's mn entries need be examined, which makes the algorithm especially useful when the cost array is given not as an explicit list of mn entries but rather in some compact implicit form. Hoffman's work implies that this northwest-corner-rule greedy algorithm solves the transportation problem if and only if the underlying cost array satisfies

$$c[i_1, j_1] + c[i_2, j_2] \leq c[i_1, j_2] + c[i_2, j_1] \quad (1)$$

for all rows i_1 and i_2 such that $1 \leq i_1 < i_2 \leq m$ and all columns j_1 and j_2 such that $1 \leq j_1 < j_2 \leq n$.

The above property is commonly known as the *Monge property*¹. It is named for the French mathematician Gaspard Monge, who in the eighteenth century exploited this property in solving certain problems involving the transportation of earth [18]. The Monge property was rediscovered in 1963 by Hoffman (who gave the property its name) and has recently drawn renewed interest in such diverse areas as coding theory [5, 16, 21], computational biology [12, 17], operations research [4], computational geometry [1, 2, 3], statistics [13], and the theory of greedy algorithms [6, 7].

The classical two-dimensional transportation problem has a natural generalization to higher dimensions. The d -dimensional transportation problem's inputs are d supply-demand vectors A_1, \dots, A_d , where the k th vector $A_k = \{a_k[i]\}$ has n_k entries, and an $n_1 \times n_2 \times \dots \times n_d$ cost array $C = \{c[i_1, i_2, \dots, i_d]\}$, such that the entries of A_1, A_2, \dots, A_d are all positive and $\sum_i a_1[i] = \sum_i a_2[i] = \dots = \sum_i a_d[i]$. The problem's objective is then to choose an $n_1 \times n_2 \times \dots \times n_d$ variable array $X = \{x[i_1, i_2, \dots, i_d]\}$ in order to

$$\begin{aligned} & \text{minimize} && \sum_{i_1, i_2, \dots, i_d} c[i_1, i_2, \dots, i_d] x[i_1, i_2, \dots, i_d] \\ & \text{subject to} && \sum_{\substack{i_1, i_2, \dots, i_d \\ \text{s.t. } i_k = I}} x[i_1, i_2, \dots, i_d] = a_k[I] && \text{for } 1 \leq k \leq d \text{ and } 1 \leq I \leq n_k \\ & && x[i_1, i_2, \dots, i_d] \geq 0 && \text{for all } i_1, i_2, \dots, i_d. \end{aligned}$$

¹Some researchers — see Burkard [10] and Cechlárová and Szabó [11], for example — call (1) the *strong Monge property* in order to distinguish it from another weaker property of square (i.e., $n \times n$) arrays that is also sometimes called the Monge property. This other Monge property characterizes the class of assignment problems that are solved by the identity permutation; it requires

$$c[i, i] + c[j, k] \leq c[i, k] + c[j, i]$$

for all i, j , and k such that $1 \leq i < j, k \leq n$.

One of the origins of the d -dimensional transportation problem is the following problem from statistics. We are given d (not necessarily distinct) populations from which a sample is to be drawn. This sample will be a d -tuple of units, one from each population. We are also given a probability distribution for each population which specifies the probability of selection for each population unit. Finally, we are given a cost for each possible sample. Given these inputs, we want to find an assignment of joint probabilities to the samples such that these joint probabilities are consistent with the individual probability distributions for each population, and at the same time the expected cost of a sample is minimized. This problem arises in the integration of surveys and in controlled selection [13]. It is not hard to see that this problem of assigning joint probabilities is equivalent to the d -dimensional transportation problem; the probability distributions of the populations correspond to the supply-demand vectors A_1, \dots, A_d , the samples' costs correspond to the cost array C , and the joint probabilities correspond to the variable array X .

Using linear programming, the d -dimensional transportation problem can be solved in time polynomial in $\prod_{k=1}^d n_k$, the size of the problem's input. In this paper, we identify conditions under which an $O(d \sum_i n_k)$ -time greedy algorithm solves the d -dimensional transportation problem. Specifically, we show that the natural greedy algorithm solves an instance of the d -dimensional transportation problem if and only if the transportation problem's cost array has a d -dimensional Monge property recently proposed by Aggarwal and Park [2, 3]. They studied this property because of certain monotonicity properties it implies. Our primary contribution is proving that Aggarwal and Park's notion of a d -dimensional Monge array is the "right" generalization of Hoffman's notion of a two-dimensional Monge array, at least insofar as the transportation problem is concerned. This paper broadens the results obtained by Bein, Brucker, and Pathak [8], who identified a smaller class of cost arrays that allow the d -dimensional transportation problem to be solved in a greedy fashion.

Note that if the cost array associated with an instance of the d -dimensional transportation problem is given in some compact implicit form, so that the problem's input has size $\Theta(\sum_{k=1}^d n_k)$ rather than $\Theta(\prod_{k=1}^d n_k)$, then any algorithm examining every entry of the cost array at least once requires time exponential in the problem's input size. However, the greedy algorithm runs in $O(d \sum_{k=1}^d n_k)$ time, which is still polynomial in the input size.

The remainder of this paper is organized as follows. Section 2 presents the property characterizing the class of d -dimensional Monge arrays. We also identify several additional properties of such arrays and argue that the class of d -dimensional Monge arrays is quite rich. In particular, we show that Monge arrays are closely related to the large class of distribution arrays. Section 3 reviews Hoffman's result for the classical transportation problem, and then Section 4 describes our if-and-only-if result for the d -dimensional transportation problem. Finally, Section 5 gives a few concluding remarks.

2 A Higher Dimensional Monge Property

This section defines the class of d -dimensional Monge arrays. We also present several useful properties of such arrays and give several examples.

Definition 2.1 For $d \geq 2$, an $n_1 \times n_2 \times \dots \times n_d$ d -dimensional array $C = \{c[i_1, i_2, \dots, i_d]\}$ has the *Monge property* if for all entries $c[i_1, i_2, \dots, i_d]$ and $c[j_1, j_2, \dots, j_d]$, we have

$$c[s_1, s_2, \dots, s_d] + c[t_1, t_2, \dots, t_d] \leq c[i_1, i_2, \dots, i_d] + c[j_1, j_2, \dots, j_d],$$

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

Note that the above definition is consistent with the two-dimensional Monge property. We will call an array a *Monge* array if it possesses the Monge property.

The following lemma, due to Aggarwal and Park, gives an equivalent definition for the class of d -dimensional Monge arrays.

Lemma 2.2 (Aggarwal and Park [2, 3]) An $n_1 \times n_2 \times \cdots \times n_d$ d -dimensional array $C = \{c[i_1, i_2, \dots, i_d]\}$ is Monge if and only if every two-dimensional plane of C is Monge. ■

Aggarwal and Park were interested in d -dimensional Monge arrays because these arrays possess the following monotonicity property. For $d \geq 2$, let $C = \{c[i_1, i_2, \dots, i_d]\}$ denote an $n_1 \times n_2 \times \cdots \times n_d$ array, and let $f_2(I_1), \dots, f_d(I_1)$ denote the second through d th coordinates of the minimum entry in the $(d-1)$ -dimensional plane consisting of those entries whose first coordinate is I_1 , so that

$$c[I_1, f_2(I_1), \dots, f_d(I_1)] = \min_{i_2, \dots, i_d} c[I_1, i_2, \dots, i_d].$$

(If the plane corresponding to I_1 contains multiple minima, we chose the first of the minima ordered lexicographically by their second through d th coordinates.) If C is Monge, then for all I_1 and J_1 satisfying $1 \leq I_1 < J_1 \leq n_1$, we clearly must have $f_k(I_1) \leq f_k(J_1)$ for all k in the range $2 \leq k \leq d$.

The set of d -dimensional Monge arrays is quite rich. One broad class of Monge arrays is motivated from statistics.

Definition 2.3 For $d \geq 2$, an $n_1 \times n_2 \times \cdots \times n_d$ d -dimensional array $C = \{c[i_1, i_2, \dots, i_d]\}$ is a *distribution array* if

$$c[i_1, i_2, \dots, i_d] = \sum_{\substack{\ell_1, \ell_2, \dots, \ell_d \\ \text{s.t. } 1 \leq \ell_k \leq i_k \\ \text{for } 1 \leq k \leq d}} p[\ell_1, \ell_2, \dots, \ell_d]$$

where $p[\ell_1, \ell_2, \dots, \ell_d] \leq 0$ for all $\ell_1, \ell_2, \dots, \ell_d$ ².

Theorem 2.4 Every distribution array is a Monge array.

Proof Consider d -tuples (i_1, i_2, \dots, i_d) , (j_1, j_2, \dots, j_d) , (s_1, s_2, \dots, s_d) , and (t_1, t_2, \dots, t_d) , as in Definition 2.1. Let $I = \{(\ell_1, \ell_2, \dots, \ell_d) \mid 1 \leq \ell_k \leq i_k \text{ for } 1 \leq k \leq d\}$. Similarly, define J , S , and T for (j_1, j_2, \dots, j_d) , (s_1, s_2, \dots, s_d) , and (t_1, t_2, \dots, t_d) , respectively. Clearly, $S = I \cap J$ and $T \supset I \cup J$. Moreover, if we define

$$F(I) = \sum_{(\ell_1, \ell_2, \dots, \ell_d) \in I} p[\ell_1, \ell_2, \dots, \ell_d]$$

(and similarly, $F(J)$, $F(S)$, and $F(T)$), we find

$$\begin{aligned} c[i_1, i_2, \dots, i_d] + c[j_1, j_2, \dots, j_d] &= F(I) + F(J) \\ &= F(I \cap J) + F(I \cup J) \\ &\geq F(S) + F(T) \\ &= c[s_1, s_2, \dots, s_d] + c[t_1, t_2, \dots, t_d], \end{aligned}$$

as we needed to show. ■

²Distribution arrays are usually defined with nonnegative $p[\ell_1, \ell_2, \dots, \ell_d]$; however, it is more natural to work with nonpositive $p[\ell_1, \ell_2, \dots, \ell_d]$ when considering with Monge arrays.

As for the converse, not all Monge arrays are distribution arrays. However, two-dimensional Monge arrays and distribution arrays are closely related, as outlined in the following.

Definition 2.5 Let $C = \{c[i_1, i_2, \dots, i_d]\}$ denote an $n_1 \times n_2 \times \dots \times n_d$ d -dimensional array. We recursively define an array function $\Delta_d c[i_1, i_2, \dots, i_d]$ as follows. If $d = 1$, then

$$\Delta_1 c[i_1] = c[i_1] - c[i_1 - 1] .$$

If $d \geq 2$, then

$$\Delta_d c[i_1, i_2, \dots, i_d] = \Delta_{d-1} c_{i_d}[i_1, i_2, \dots, i_{d-1}] - \Delta_{d-1} c_{i_d-1}[i_1, i_2, \dots, i_{d-1}] ,$$

where $C_{i_d} = \{c_{i_d}[i_1, i_2, \dots, i_{d-1}]\}$ denotes the $(d-1)$ -dimensional subarray of C corresponding to a fixed value i_d of C 's last index. By convention, $c[i_1, i_2, \dots, i_d] = 0$ if any of i_1, i_2, \dots, i_d is 0.

For a nonrecursive definition of $\Delta_d c[i_1, i_2, \dots, i_d]$, let

$$L(i_1, i_2, \dots, i_d) = \{(\ell_1, \ell_2, \dots, \ell_d) \mid \ell_k = i_k - 1 \text{ or } \ell_k = i_k \text{ for } 1 \leq k \leq d\} ,$$

$$L^+(i_1, i_2, \dots, i_d) = \{(\ell_1, \ell_2, \dots, \ell_d) \in L(i_1, i_2, \dots, i_d) \mid \ell_k = i_k - 1 \text{ an even number of times}\} ,$$

and

$$L^-(i_1, i_2, \dots, i_d) = \{(\ell_1, \ell_2, \dots, \ell_d) \in L(i_1, i_2, \dots, i_d) \mid \ell_k = i_k - 1 \text{ an odd number of times}\} .$$

Then

$$\Delta_d c[i_1, i_2, \dots, i_d] = \sum_{\ell_1, \dots, \ell_d \in L^+(i_1, \dots, i_d)} c[\ell_1, \dots, \ell_d] - \sum_{\ell_1, \dots, \ell_d \in L^-(i_1, \dots, i_d)} c[\ell_1, \dots, \ell_d] .$$

Theorem 2.6 For $d \geq 2$, an $n_1 \times n_2 \times \dots \times n_d$ d -dimensional array $C = \{c[i_1, i_2, \dots, i_d]\}$ is a distribution array if and only if $\Delta_d c[i_1, i_2, \dots, i_d] \leq 0$ for all entries $c[i_1, i_2, \dots, i_d]$ in C .

Proof All we have to show is that

1. every d -dimensional array $C = \{c[i_1, i_2, \dots, i_d]\}$ has a unique representation

$$c[i_1, i_2, \dots, i_d] = \sum_{\substack{\ell_1, \ell_2, \dots, \ell_d \\ \text{s.t. } 1 \leq \ell_k \leq i_k \\ \text{for } 1 \leq k \leq d}} p[\ell_1, \ell_2, \dots, \ell_d] ,$$

and

- 2.

$$c[i_1, i_2, \dots, i_d] = \sum_{\substack{\ell_1, \ell_2, \dots, \ell_d \\ \text{s.t. } 1 \leq \ell_k \leq i_k \\ \text{for } 1 \leq k \leq d}} \Delta_d c[\ell_1, \ell_2, \dots, \ell_d] .$$

The uniqueness is easily proven using a simple induction over $i_1 + i_2 + \dots + i_d$, whereas the second equation follows directly from the recursive definition of $\Delta_d c[i_1, i_2, \dots, i_d]$. ■

In two dimensions, $\Delta_2 c[i, j] = c[i, j] + c[i - 1, j - 1] - c[i, j - 1] - c[i - 1, j] \leq 0$ for all i and j implies that C is a Monge array and also that C is nonpositive and nonincreasing in its rows and columns. (The sign and monotonicity conditions come from the cases where $i = 1$ or $j = 1$.) This observation gives a nice characterization of two-dimensional Monge arrays in terms of distribution arrays, first noted by Bein and Pathak.

Theorem 2.7 (Bein and Pathak [9]) A two-dimensional array $C = \{c[i, j]\}$ is a Monge array if and only if there exists a distribution array $D = \{d[i, j]\}$ and vectors $U = \{u[i]\}$ and $V = \{v[j]\}$ such that

$$c[i, j] = u[i] + v[j] + d[i, j] .$$

Proof The “if” direction of the proof is immediate, since by Theorem 2.4, D is a Monge array, and since U and V cancel out in the Monge property.

For the “if only” direction, consider any Monge array $C = \{c[i, j]\}$. We can make C nonpositive and nonincreasing in its rows and columns as follows. Define $D = \{d[i, j]\}$ so that $d[i, j] = c[i, j] - c[i, 1] - c[1, j] + c[1, 1]$ and thus obtain $c[i, j] = u[i] + v[j] + d[i, j]$, where $u[i] = c[i, 1]$ and $v[j] = c[1, j] - c[1, 1]$. The array D is Monge, nonpositive, and nonincreasing in its rows and columns; hence, $\Delta_2 d[i, j] \leq 0$ for all i and j , which implies D is a distribution array by Theorem 2.6. ■

We close this section with a few examples of d -dimensional arrays $C = \{c[i_1, i_2, \dots, i_d]\}$ that are Monge:

1. $c[i_1, i_2, \dots, i_d] = i_1 + i_2 + \dots + i_d$,
2. $c[i_1, i_2, \dots, i_d] = -(i_1 + i_2 + \dots + i_d)^2$,
3. $c[i_1, i_2, \dots, i_d] = \max\{i_1, i_2, \dots, i_d\}$, and
4. $c[i_1, i_2, \dots, i_d] = \max\{i_1, i_2, \dots, i_d\} - \min\{i_1, i_2, \dots, i_d\}$.

Note that the second array defined above is a distribution array and that each of the remaining three arrays can be transformed into a distribution array by subtracting an appropriate constant from each entry of the array. Also note that since every subarray of a Monge array is Monge, arrays such as $c[i_1, i_2, \dots, i_d] = \max\{a_1[i_1], a_2[i_2], \dots, a_d[i_d]\}$ are also Monge, provided the vectors $A_1 = \{a_1[i_1]\}$ through $A_d = \{a_d[i_d]\}$ are nondecreasing, i.e., $a_k[1] \leq a_k[2] \leq \dots \leq a_k[n_k]$ for $1 \leq k \leq d$.

3 Hoffman’s Result for the Classical Transportation Problem

This section reviews a special case of Hoffman’s result [15] for the classical two-dimensional transportation problem. Specifically, we show that a natural greedy algorithm solves the two-dimensional transportation problem for a particular cost array if and only if the cost array has the two-dimensional Monge property. We include a detailed description of this result because its proof parallels the proof of our if-and-only-if result for the d -dimensional transportation problem, which we present in Section 4.

Hoffman’s original result is somewhat more general than the result we are going to prove in this section. In particular, his result for the two-dimensional transportation problem is stated in terms of Monge sequences, not Monge arrays. A *Monge sequence* σ for an $m \times n$ array $C = \{c[i, j]\}$ is an

ordering of the mn entries of C (or, equivalently, the transportation problem's mn variables) with the following property: for all rows i and k and all columns j and ℓ , if $c[i, j]$ precedes $c[i, \ell]$ and $c[j, k]$ with respect to σ , then

$$c[i, j] + c[k, \ell] \leq c[i, \ell] + c[k, j]. \quad (2)$$

In other words, σ is a one-to-one function mapping $\{1, \dots, m\} \times \{1, \dots, n\}$ to $\{1, \dots, mn\}$ such that for all i, j, k , and ℓ , $\sigma(i, j) < \sigma(i, \ell)$ and $\sigma(i, j) < \sigma(k, j)$ together imply (2). Note that an $m \times n$ array C is a Monge array if and only if the sequence $\sigma(i, j) = (i-1)n + j$ is a Monge sequence for C . Also note that it is possible to have a Monge sequence for a non-Monge array.

Given a particular sequence σ , Hoffman considered the following $O(mn)$ -time greedy algorithm, which we will call GREEDY_σ . The algorithm begins by assigning a value to the “first” variable $x[i, j]$, i.e., the variable corresponding to the pair (i, j) such that $\sigma(i, j) = 1$. Specifically, it sets $x[i, j] = \min\{a[i], b[j]\}$ and reduces both $a[i]$ and $b[j]$ by $\min\{a[i], b[j]\}$ (to reflect the supply and demand “used up” by this assignment). The algorithm then proceeds to the “second” variable $x[i', j']$ (according to σ), setting $x[i', j'] = \min\{a[i'], b[j']\}$ and reducing both $a[i']$ and $b[j']$ by $\min\{a[i'], b[j']\}$. The algorithm continues in this manner, processing the variables one at a time in the order given by σ , until all the variables have received values, all the supply has been exhausted, and all the demand has been satisfied. Note that GREEDY_σ always selects a feasible solution for the transportation problem and that its running time is $O(mn)$ (since the algorithm spends constant time processing each variable).

The solution selected by GREEDY_σ has a nice characterization in terms of the total order that σ induces on the set of all solutions. A solution is defined by its vector of variables $(x[1, 1], x[1, 2], \dots, x[m, n-1], x[m, n])$; reordering the variables in these vectors according to σ (so that $x[i, j]$ appears in position $\sigma(i, j)$ of the vector) and then ordering these vectors lexicographically gives an ordering of all solutions. A solution $X = \{x[i, j]\}$ then follows another solution $X' = \{x'[i, j]\}$ with respect to σ (which we will denote $X \succ X'$) if and only if there exists a pair (i, j) such that $x[i, j] > x'[i, j]$ and $x[i', j'] = x'[i', j']$ for all (i', j') such that $\sigma(i', j') < \sigma(i, j)$. It is not hard to see that the algorithm GREEDY_σ selects the feasible solution that is lexicographically last with respect to σ .

As his main result, Hoffman proved that GREEDY_σ solves the two-dimensional transportation problem for a particular cost array C and all vectors A and B if and only if σ is a Monge sequence for C . In other words, if a cost array C has a Monge sequence σ , then, for any A and B , the last feasible solution with respect to σ is an optimal solution.

The main focus of this section is the special case of Hoffman's result where σ is given by $\sigma(i, j) = (i-1)n + j$ (i.e., the cost array is Monge). In this case, the algorithm GREEDY_σ can be modified to run in $O(m+n)$ time, since every time a variable's value is fixed, either one row or one column of C can be eliminated, and it is easy to update σ to reflect this elimination (so that eliminated entries are not processed). We will call this modified algorithm GREEDY_2 .

The algorithm GREEDY_2 can be viewed as following a “northwest corner rule.” If we imagine embedding the array X in the plane, so that $x[1, 1]$ lies in the northwest corner of the array and $x[m, n]$ lies in the southeast corner, then GREEDY_2 works by assigning a value to the north- and westmost variable, which eliminates either the northmost row or westmost column (or both), and then recursively solving the transportation problem that remains.

We will now prove that GREEDY_2 works if and only if C is Monge. This result follows as a special case of Hoffman's more general result; however, we prove it directly here because we will use a similar approach to prove our result for the d -dimensional transportation problem in the next section.

Theorem 3.1 (Hoffman [15]) Given a particular $m \times n$ two-dimensional cost array C , the algorithm GREEDY_2 solves the corresponding two-dimensional transportation problem for all A and B if and only if C is Monge.

Proof The “only if” direction of the proof is quite simple. Suppose C is not Monge. This assumption implies the existence of entries $c[i_1, j_1]$ and $c[i_2, j_2]$ such that $1 \leq i_1 < i_2 \leq m$, $1 \leq j_1 < j_2 \leq n$, and

$$c[i_1, j_1] + c[i_2, j_2] > c[i_1, j_2] + c[i_2, j_1] .$$

Now consider the vectors A and B where $a[i_1] = a[i_2] = b[j_1] = b[j_2] = 1$ and all other entries in both A and B are zero. Clearly,

$$\sum_{i=1}^m a[i] = \sum_{j=1}^n b[j] .$$

Moreover, given these particular A and B as inputs, the greedy algorithm sets $x[i_1, j_1]$ and $x[i_2, j_2]$ to 1 (and all other variables to 0). However, setting $x[i_1, j_2]$ and $x[i_2, j_1]$ to 1 (and all other variables to 0) is also feasible and, by assumption, its cost $c[i_1, j_2] + c[i_2, j_1]$ is strictly less than the cost $c[i_1, j_1] + c[i_2, j_2]$ of the variable assignment selected by the greedy algorithm. Hence, the greedy algorithm does not solve the transportation problem for all A and B .

For the “if” direction of the proof, suppose the lexicographically last feasible solution X is not optimal, and let X' denote the lexicographically last solution that is optimal. By assumption, $c(X) > c(X')$ and $X \succ X'$. Now consider the lexicographically first variable $x[i, j]$ to which X and X' assign different values. Since X follows X' in the lexicographic ordering of solutions, we must have $x[i, j] > x'[i, j]$. Thus, in order to insure that the supply $a[i]$ and the demand $b[j]$ are both satisfied, X' must assign nonzero values to at least two distinct variables $x'[i, s]$ and $x'[r, j]$ such that $i < r \leq m$ and $j < s \leq n$. However, since C is Monge, we have

$$c[i, j] + c[r, s] \leq c[i, s] + c[r, j] .$$

Thus, reducing $x'[i, s]$ and $x'[r, j]$ by $\epsilon = \min\{x'[i, s], x'[r, j]\}$ and increasing $x'[i, j]$ and $x'[r, s]$ by this same ϵ gives a new feasible solution X'' such that $c(X'') \leq c(X')$ and $X'' \succ X'$. The existence of X'' contradicts our assumption that X' is the lexicographically last optimal solution. ■

4 Our Result for the d -Dimensional Transportation Problem

In this section, we present our if-and-only-if result for the d -dimensional transportation problem.

We begin by defining our greedy algorithm. This algorithm, which we will call GREEDY_d , is the natural extension of the northwest-corner-rule greedy algorithm GREEDY_2 described in the previous section. Our algorithm begins by setting $x[1, 1, \dots, 1] = \min\{a_1[1], a_2[1], \dots, a_d[1]\}$. We then reduce each of $a_1[1], a_2[1], \dots, a_d[1]$ by $\min\{a_1[1], a_2[1], \dots, a_d[1]\}$. At least one of $a_1[1], a_2[1], \dots, a_d[1]$ is thereby reduced to 0, which means we are done with the corresponding $(d-1)$ -dimensional subarray of X . In other words, at least one of the problem’s dimensions n_1, n_2, \dots, n_d has been reduced by 1. The smaller transportation problem that remains can then be solved recursively.

The algorithm GREEDY_d again selects the lexicographically last feasible solution, and its running time is $O(d(n_1 + n_2 + \dots + n_d))$, since each variable assignment takes $O(d)$ time and reduces at least one of the problem’s dimensions by 1.

Theorem 4.1 Given a particular $n_1 \times n_2 \times \dots \times n_d$ d -dimensional cost array C , the algorithm GREEDY_d solves the corresponding d -dimensional transportation problem for all A_1, A_2, \dots, A_d if and only if C is Monge.

Proof The “only if” direction of the proof is again quite simple. Suppose C is not Monge. This assumption implies the existence of entries $c[i_1, i_2, \dots, i_d]$ and $c[j_1, j_2, \dots, j_d]$ such that

$$c[s_1, s_2, \dots, s_d] + c[t_1, t_2, \dots, t_d] > c[i_1, i_2, \dots, i_d] + c[j_1, j_2, \dots, j_d],$$

where $s_k = \min\{i_k, j_k\}$ and $t_k = \max\{i_k, j_k\}$ for $1 \leq k \leq d$. Note that the d -tuples (i_1, i_2, \dots, i_d) , (j_1, j_2, \dots, j_d) , (s_1, s_2, \dots, s_d) , and (t_1, t_2, \dots, t_d) must be distinct. Now consider the vectors A_1, A_2, \dots, A_d where for $1 \leq k \leq d$, we have $a_k[i_k] = 1$ and $a_k[j_k] = 1$ if $i_k \neq j_k$ and $a_k[i_k] = 2$ if $i_k = j_k$, and all other entries in A_1, A_2, \dots, A_d are zero. Clearly,

$$\sum_{i=1}^{n_1} a_1[i] = \sum_{i=1}^{n_2} a_2[i] = \dots = \sum_{i=1}^{n_d} a_d[i] = 2.$$

Moreover, given these particular A_1, A_2, \dots, A_d as inputs, the greedy algorithm sets $x[s_1, s_2, \dots, s_d]$ and $x[t_1, t_2, \dots, t_d]$ to 1 (and all other variables to 0). However, the solution setting $x[i_1, i_2, \dots, i_d]$ and $x[j_1, j_2, \dots, j_d]$ to 1 (and all other variables to 0) is also feasible, and, by assumption, its cost $c[i_1, i_2, \dots, i_d] + c[j_1, j_2, \dots, j_d]$ is strictly less than the cost $c[s_1, s_2, \dots, s_d] + c[t_1, t_2, \dots, t_d]$ of the solution selected by the greedy algorithm. Hence, the greedy algorithm does not solve the transportation problem for all A_1, A_2, \dots, A_d .

For the “if” direction of the proof, suppose the lexicographically last feasible solution X is not optimal, and let X' denote the lexicographically last solution that is optimal. By assumption, $c(X) > c(X')$ and $X \succ X'$. Now consider the lexicographically first variable $x[i_1, i_2, \dots, i_d]$ to which X and X' assign different values. Since X follows X' in the lexicographic ordering of solutions, we must have $x[i_1, i_2, \dots, i_d] > x'[i_1, i_2, \dots, i_d]$. Thus, in order to insure that the $a_1[i_1], a_2[i_2], \dots, a_d[i_d]$ are all satisfied, X' must assign nonzero values to d variables

$$\begin{aligned} & x'[j_1^1, j_2^1, \dots, j_d^1] \\ & x'[j_1^2, j_2^2, \dots, j_d^2] \\ & \vdots \\ & x'[j_1^d, j_2^d, \dots, j_d^d] \end{aligned}$$

such that for $1 \leq k \leq d$ and $1 \leq \ell \leq d$, $j_\ell^k \geq i_\ell$, and for $1 \leq k \leq d$, $j_k^k = i_k$. All d of the d -tuples $(j_1^k, j_2^k, \dots, j_d^k)$ need not be distinct, but there must exist at least two of them, say $(j_1^v, j_2^v, \dots, j_d^v)$ and $(j_1^w, j_2^w, \dots, j_d^w)$, neither of which dominates the other (i.e., there exist k and ℓ such that $j_k^v < j_k^w$ and $j_\ell^v > j_\ell^w$). However, since C is Monge, we have

$$c[s_1, s_2, \dots, s_d] + c[t_1, t_2, \dots, t_d] \leq c[j_1^v, j_2^v, \dots, j_d^v] + c[j_1^w, j_2^w, \dots, j_d^w],$$

where $s_k = \min\{j_k^v, j_k^w\}$ and $t_k = \max\{j_k^v, j_k^w\}$ for $1 \leq k \leq d$. Furthermore, since neither of the d -tuples $(j_1^v, j_2^v, \dots, j_d^v)$ and $(j_1^w, j_2^w, \dots, j_d^w)$ dominates the other, the d -tuples (s_1, s_2, \dots, s_d) and (t_1, t_2, \dots, t_d) must be distinct from $(j_1^v, j_2^v, \dots, j_d^v)$ and $(j_1^w, j_2^w, \dots, j_d^w)$. Thus, reducing $x'[j_1^v, j_2^v, \dots, j_d^v]$ and $x'[j_1^w, j_2^w, \dots, j_d^w]$ by

$$\epsilon = \min\{x'[j_1^v, j_2^v, \dots, j_d^v], x'[j_1^w, j_2^w, \dots, j_d^w]\}$$

and increasing $x'[s_1, s_2, \dots, s_d]$ and $x'[t_1, t_2, \dots, t_d]$ by this same ϵ gives a new feasible solution X'' such that $c(X'') \leq c(X')$ and $X'' \succ X'$. The existence of X'' contradicts our assumption that X' is the lexicographically last optimal solution. ■

5 Concluding Remarks

We conclude with two remarks:

1. Closely related to the d -dimensional transportation problem is the d -dimensional assignment (or matching) problem. The assignment problem is a transportation problem where
 - (a) all the dimensions have equal size (i.e., $n_k = n$ for $1 \leq k \leq d$),
 - (b) all the supply-demands are 1 (i.e., $a_k[i_k] = 1$ for $1 \leq k \leq d$ and $1 \leq i_k \leq n$), and
 - (c) the variables $x[i_1, i_2, \dots, i_d]$ are restricted to be integers.

For $d \geq 3$, the general d -dimensional assignment problem is NP-hard (see Garey and Johnson [14], for example). However, if the assignment problem's cost array is Monge, then the solution given by

$$x[i_1, i_2, \dots, i_d] = \begin{cases} 1 & \text{if } i_1 = i_2 = \dots = i_d, \\ 0 & \text{otherwise} \end{cases}$$

must be optimal, as our greedy algorithm will always produce this solution.

2. In this paper, we have focused on generalizing the Monge-array formulation of Hoffman's work. A natural question to ask is whether Hoffman's notion of a Monge sequence also generalizes to higher dimensions. We believe that it does, though the property defining a d -dimensional Monge sequence seems to be a bit more complicated than the property defining a d -dimensional Monge array. (Specifically, the Monge-sequence property relates $2d$ entries of the cost array, while the Monge-array property relates only 4 entries.) However, we have not pursued this line of investigation, as the natural Monge-sequence greedy algorithm takes $O(n^d)$ time to solve an $n \times n \times \dots \times n$ d -dimensional transportation problem, making the Monge-sequence case of somewhat less interest than the Monge-array case, where the greedy algorithm runs in $O(d^2n)$ time.

References

- [1] A. Aggarwal, M. M. Klawe, S. Moran, P. W. Shor, and R. Wilber. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 2(2):195–208, 1987.
- [2] 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 *Journal of Algorithms*. Portions of this paper appear in *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 497–512, 1988.
- [3] 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 *Journal of Algorithms*. Portions of this paper appear in *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 497–512, 1988.
- [4] A. Aggarwal and J. K. Park. Improved algorithms for economic lot-size problems. *Operations Research*, 1992. To appear. An earlier version of this paper appears as Research Report RC 15626, IBM T. J. Watson Research Center, Yorktown Heights, NY, March 1990.

- [5] M. J. Atallah, S. R. Kosaraju, L. L. Larmore, G. Miller, and S. Teng. Constructing trees in parallel. In *Proceedings of the 1st Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 421–431, 1989.
- [6] W. W. Bein. *Netflows, Polymatroids, and Greedy Structures*. PhD thesis, Universität Osnabrück, Osnabrück, Germany, 1986.
- [7] W. W. Bein, P. Brucker, and A. J. Hoffman. Series parallel composition of greedy linear programming problems. Research Report RC 17834, IBM T. J. Watson Research Center, Yorktown Heights, NY, February 1992. Submitted to *Mathematical Programming*.
- [8] W. W. Bein, P. Brucker, and P. K. Pathak. Monge properties in higher dimensions. Technical Report CS90-11, Department of Computer Science, University of New Mexico, Albuquerque, NM, October 1990.
- [9] W. W. Bein and P. K. Pathak. A characterization of the Monge property. Technical Report CS90-10, Department of Computer Science, University of New Mexico, Albuquerque, NM, September 1990.
- [10] R. E. Burkard. Assignment problems: Recent solution methods and applications. In A. Prékopa, J. Szelezsán, and B. Strazicky, editors, *System Modelling and Optimization: Proceedings of 12th IFIP Conference, Budapest, Hungary, September 2–6, 1985*, volume 84 of *Lecture Notes in Control and Information Sciences*, pages 153–169. Springer-Verlag, New York, NY, 1986.
- [11] K. Cechlárová and P. Szabó. On the Monge property of matrices. *Discrete Mathematics*, 81(2):123–128, 1990.
- [12] D. Eppstein, Z. Galil, R. Giancarlo, and G. F. Italiano. Sparse dynamic programming. In *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 513–522, 1990.
- [13] M. Fahimi and P. K. Pathak. Applications of a size reduction technique for transportation problems in sampling. In *Proceedings of the First International Symposium on Optimization and Statistics, Aligarh, India*, December 1989.
- [14] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco, CA, 1979.
- [15] A. J. Hoffman. On simple linear programming problems. In V. Klee, editor, *Convexity: Proceedings of the Seventh Symposium in Pure Mathematics of the AMS*, volume 7 of *Proceedings of Symposia in Pure Mathematics*, pages 317–327. American Mathematical Society, Providence, RI, 1963.
- [16] L. L. Larmore and T. M. Przytycka. Parallel construction of trees with optimal weighted path length. In *Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 71–80, 1991.
- [17] L. L. Larmore and B. Schieber. On-line dynamic programming with applications to the prediction of RNA secondary structure. *Journal of Algorithms*, 12(3):490–515, 1991.

- [18] G. Monge. Mémoire sur la théorie des déblais et des remblais. In *Histoire de l'Académie Royale des Sciences, Année M. DCCLXXXI, avec les Mémoires de Mathématique et de Physique, pour la même Année, Tirés des Registres de cette Académie*, pages 666–704. L'Imprimerie Royale, Paris, France, 1784.
- [19] J. B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 1992. To appear. An earlier version of this paper appears in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 377–387, 1988.
- [20] H. M. Wagner. On a class of capacitated transportation problems. *Management Science*, 5(3):304–318, 1959.
- [21] F. F. Yao. Efficient dynamic programming using quadrangle inequalities. In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing*, pages 429–435, 1980.