Consiglio Nazionale delle Ricerche

Istituto di Elaborazione dell'Informazione

A general class of greedily solvable Linear Programs

Fabio Tardella
Maurice Queyranne
Frits Spieksma

Nota Interna IEI-B4-49
ottobre 1992

Via S. Maria, 46 - 56126 Pisa (Italy)
Phone : +39.(0)50.553159
Fax : +39.(0)50.554342
Telex : 590305 IEICNR I

A General Class of Greedily Solvable Linear Programs

Maurice Queyranne
University of British Columbia, Vancouver, Canada

Frits Spieksma
University of Limburg, Maastricht, The Netherlands

and Fabio Tardella
Istituto d'Elaborazione dell' Informazione, Pisa, Italy

Extended Abstract
September 1992

Abstract

A greedy algorithm solves a dual pair of linear programs where the primal variables are associated to the elements of a sublattice B of a finite product lattice, and the cost coefficients define a submodular function on B. This approach unifies and generalizes two well-known classes of greedily solvable linear programs. The primal problem generalizes the (ordinary and multi-index) transportation problems satisfying a Monge condition (Hoffman, 1963; Bein et al., 1992) to the case of forbidden cells. The dual problem generalizes the linear optimization problem over submodular polyhedra (Lovász, 1983; Fujishige and Tomizawa, 1983), which stemmed from the work of Edmonds (1970) on polymatroids to an arbitrary finite product lattice. We also discuss relationships between Monge properties and submodularity, and present a class of problems with submodular costs arising in production and logistics.

1 Introduction.

An important class of simple and efficient algorithms for optimizing a function f on a set S is the class of *greedy* (or myopic) algorithms. Since the work of Edmonds [14], [15] on matroids and of Hoffman [24] on transportation problems, numerous authors have studied conditions on f and S which guarantee the convergence of greedy algorithms to optimal solutions.

In the case where f is linear and S is a polyhedron, two broad and well-known classes of linear programs have been shown to be optimally solvable by a greedy algorithm. Following the work of Edmonds, the first class includes optimization problems on polymatroids and related submodular polyhedra, see Frank and Tardos [17] and Fujishige's monograph [19] for in-depth studies. On the other hand, following the work of Hoffman, the second class includes transportation problems, both ordinary and multi-index, with cost coefficients satisfying some form of a so-called *Monge condition* (see Hoffman [26] and Bein et al. [11] for details).

In this paper, we present a dual pair of linear programs, in which the variables are associated with the elements in a sublattice of a discrete product lattice. We show that a greedy algorithm solves both primal and dual programs when the cost coefficients in the primal problem (or, equivalently, the right hand sides in the dual problem) are given by a submodular function on the sublattice. The primal problem generalizes the multi-index transportation problem of Bein et al. [11] to the case of forbidden arcs with a sublattice structure. The dual problem generalizes the linear optimization problems on submodular polyhedra by Lovász [30] and Fujishige and Tomizawa [20] to a distributive sublattice of a finite product space.

In addition to enlarging the class of linear programs solvable by a greedy algorithm, our work also unifies heretofore separate streams of research and highlights the duality relationship between (multi-index) transportation problems and linear optimization problems on submodular polyhedra. In particular, we observe that submodularity and the Monge condition are the same concept expressed in different forms. Indeed, known results on lattices and submodular functions are independently re-discovered in the context of the Monge condition for multi-dimensional arrays. Conversely, new results for Monge matrices also apply to submodular functions.

The content of this paper is as follows. In Section 2, we define the dual pair of linear programs which is the object of this paper. We show how they generalize multi-index transportation problems and linear optimization problems on submodular polyhedra. In Section 3, we present the greedy algorithm and prove that it produces optimal solutions to the primal and the dual problems. In Section 4, we discuss relations between submodularity, the Monge condition for a matrix and the existence of Monge sequences. Finally, in Section 5, we exhibit a class of problem instances with a submodular cost function. These problem instances arise in some manufacturing and logistics environments.

2 Lattices, submodular functions, and a dual pair of linear programs.

Let the integer $k \geq 2$ denote the dimension of the product lattice defined below, and $K \doteq \{1, \ldots, k\}$. For $i \in K$, let A_i be a totally ordered set (or *chain*) with $m(i) + 1$ elements. For simplicity, we let $A_i \doteq \{0, 1, \ldots, m(i)\}$, with the usual total order $0 < 1 < \cdots < m(i)$, for all $i \in K$. The *product space* $A \doteq A_1 \times A_2 \times \cdots \times A_k$ is a distributive lattice with *join* and *meet* operations defined by

$$a \vee b \doteq \Big(\max\{a(1), b(1)\}, \ \ldots \ , \max\{a(k), b(k)\} \Big)$$

of resources). It is assumed that $\sum_{j \in A_i^*} d_{ij} = D$, a constant for all $i \in K$. With each element $a \in B$, often called an "arc", we associate a cost rate $w(a)$ and a nonnegative decision variable x_a representing the amount of "demand" which is satisfied by the corresponding combination. The (axial) k-index transportation problem is to determine the amount associated with each permissible arc $a \in B$ so as to satisfy exactly the demand of each section $B(i,j)$ (for all $i \in K$ and $j \in A_i^*$) at minimum total cost. This problem may be formulated precisely as an instance of problem (P).

The case where $B = A'$ (no forbidden arcs) is the axial multi-index transportation problem defined by Haley (p. 376 in [23]; see also Chapter 8 in [42], and [11]). The *axial k-index assignment problem* arises when all $m(i)$ are equal to a constant m all demands are equal to 1, and all variables x_a are restricted to be 0 or 1, see, e.g., Pierskalla [33] and Bandelt et al. [9]. When, in addition, $k = 3$, we have the much studied (axial) *three-dimensional assignment problem,* see Frieze and Yadegar [18], Balas and Saltzman [8], and Crama and Spieksma [13]. The above references describe several practical applications of these different models in such areas as logistics, automated production, statistics, and course scheduling.

Note that, in addition to offering a concise notation (compared with the above references), our formulation of problem (P) allows us quite naturally to exclude forbidden arcs. Ordinary (i.e., 2-index) transportation problems with forbidden arcs were considered by Shamir and Dietrich [36] in connection with the existence of Monge sequences; see Section 4 for details. Note also that we do not need to require in problem (P) that the total demand $\sum_{j \in A_i^*} d_{ij}$ be constant for all $i \in K$.

Submodular polyhedra. When $m(i) = 1$ for all $i \in K$, the lattice A may be identified with the lattice 2^K of all subsets of K. Sublattices B are then (distributive) set lattices, and submodular functions coincide with those now well-known in combinatorial optimization (see, e.g., Nemhauser and Wolsey [31]). The constraints of problem (D) are then precisely those which define a *submodular polyhedron* as in Fujishige [19] (see also Frank and Tardos [17]). When $B = A = 2^K$ we have the problem which Lovász [30] shows to be solvable by a greedy algorithm. These polyhedra are closely related to the *polymatroids* introduced by Edmonds [15], see the preceding references for details.

Problem (D) properly generalizes these submodular polyhedra by allowing each chain A_i in the lattice to contain any number of elements, giving rise to arbitrary (finite) product lattices. This is akin to extending attention, in integer programming, from binary variables to general integer-valued variables. We refer to the work of Topkis, Veinott, and Granot and Veinott cited above for a description of some of the problems amenable to this broader framework.

The Greedy Algorithm in the next section generalizes on one hand those of Hoffman and Bein et al. for (ordinary and multi-index) transportation problems, and on the other hand those in Lovász and in Fujishige and Tomizawa for submodular polyhedra (the latter two being themselves generalizations of that of Edmonds for polymatroids).

3 A greedy algorithm.

We first describe the input and output of our algorithm for problems (P) and (D). Throughout this Section, we assume that B is any sublattice of A.

Input:

integer k	the dimension of A;
integers $m(1), \ldots, m(k)$	defining the range of each coordinate of A;
reals $d_{ij} \geq 0$	demands, for all $i \in K$, $j \in A_i^*$;
oracle MAXLE	describing sublattice B (see explanations below);
oracle w	returning the value $w(b)$ for any $b \in B$.

Output:

variable *Status*	indicating the status, *Feasible* or *Infeasible*, of problem (P);

and if *Status* = *Feasible*:

list $((b^1, x_{b^1}), \ldots, (b^T, x_{b^T}))$	describing a primal solution (see below);
reals $y_{ij} \geq 0$	describing a dual solution, for all $i \in K$ and $j \in A_i^*$.

The sublattice B might be presented in different ways, such as: a list of all elements in B (the *permissible* elements or cells); a list of all elements in $A \setminus B$ (the *forbidden* elements); collections of conditions (for example, *monotone linear inequalities*, see [39]) characterizing permissible or forbidden elements; and so forth. However, to achieve sufficient generality and to exploit the intrinsic simplicity of the Greedy Algorithm, we use the following oracle, which we call MAXLE. The input to MAXLE consists of any element $a \in A$. Oracle MAXLE then returns "NULL" if the truncation B_a of B generated by a is empty; else it returns $\bigvee B_a$, the largest element of $b \in B$ such that $b \leq a$. We leave it to the interested reader which data structures can be used to efficiently implement this oracle for a given representation of the sublattice B, such as one of those outlined above

The output to the Greedy Algorithm exploits the sparseness of the basic solutions to problem (P). Although the primal solution vector x has one component x_b for every element $b \in B^*$ (and hence potentially up to $(\prod_i (m(i)+1)) - 1$ variables), at most $\sum_i m(i)$ of these will assume a positive value. Thus the Greedy Algorithm, which will be shown to produce a dual pair of basic solutions to problems (P) and (D), returns a list of T pairs (b, x_b) with $b \in B^*$ and x_b is the value of the corresponding variable in the solution. The number T of such pairs is determined by the algorithm, but will be shown not to exceed $\sum_i m(i)$. It is understood that $x_b \doteq 0$ for all $b \in B^*$ which do not appear in this list. Similarly, in the dual problem, we may set all variables y_{ij} for which $B(i,j) = \emptyset$ or $d_{ij} = 0$ to small enough, and otherwise arbitrary, values.

The Greedy Algorithm detailed below consists of two phases. The Primal Phase repeats the following step: identify (using the MAXLE oracle) the largest available element $b \in B^*$ and assign the largest possible value to its variable x_b. This step is repeated until either infeasibility is detected, in which case the algorithm halts, or all demands are satisfied. In the latter case, the list of (b, x_b) pairs output by the algorithm defines a feasible primal solution. The Dual Phase then traces back the sequence of elements $b \in B$ recorded in the Primal Phase to construct (using the w oracle) a dual solution y.

GREEDY ALGORITHM:

Primal Phase.

0. (Initialize:)
   Let $\delta_{ij} := d_{ij}$ for all i, j;

$\Lambda := K$; $a := \text{MAXLE}(m)$; t := 0;

1. (Main step):

Repeat

if ($a \neq$ NULL) then {

if (there exists (i,j) with $j > a(i)$ and $\delta_{ij} > 0$) then

return($Status := Infeasible$)

else {

$t := t + 1$;

let $i \in \text{argmin}\{\delta_{\ell,a(\ell)} : \ell \in \Lambda\}$;

$x_a := \delta_{i,a(i)}$; $\eta(t) := i$;

add $(a_t, x_{a_t}) := (a, x_a)$ to the output list;

let $\delta_{\ell,a(\ell)} := \delta_{\ell,a(\ell)} - x_a$ for all

$\ell \in \Lambda$;

$a(i) := a(i) - 1$; if $a(i) = 0$ then let

$\Lambda := \Lambda \setminus \{i\}$;

let $a := \text{MAXLE}(a)$;

}

}

until ($a = 0$ or $a =$ NULL);

2. (Final test for infeasibility):

If ($a =$ NULL and there exists (i,j) with $\delta_{ij} > 0$) then

return($Status := Infeasible$);

else { let $T := t$; output the list $(a_1, x_{a_1}), \ldots, (a_T, x_{a_T})$;

}

Dual Phase.

For $t := T$ down to 1 do

output $y_{\eta(t),a_t(\eta(t))} :=$

$$w(a_t) - \sum\{y_{\eta(u),a_t(\eta(u))} : \text{all } u > t \text{ with } a_u(\eta(u)) = a_t(\eta(u))$$

}; $Return(Status = Feasible)$.

Note that, when applied to an ordinary (two-dimensional) transportation problem, the Primal Phase reduces to the well-known North-West corner rule (with an appropriate geographic orientation of the transportation array). More generally, the Primal Phase reduces to the greedy algorithm of Bein et al. [11] for multi-index transportation problems without forbidden arcs. In the case where problem (D) is a linear optimization problem over a submodular polyhedron, (that is, $A_i = \{0,1\}$ for all $i \in K$), the Primal Phase amounts to sorting the d_{i1} values in a nondecreasing order, *consistent* with the sublattice B in the following sense: if $b(i) = 1$ for all $b \in B$ with $b(h) = 1$, and if $\eta(t) = i$ and $\eta(u) = h$, then $t > u$ (for all $h, i \in K$). Then, in the Dual Phase, the y-variables are sequentially maximized according to this sequence, with $y_{\eta(t),a_t(\eta(t))} := w(a_t) - w(a_{t+1})$ (where $w(a_{T+1} \doteq 0)$. Thus the Greedy Algorithm just presented reduces to that of Lovász [30] and of Fujishige and Tomizawa [20] in the case of submodular polyhedra.

Theorem 3.1 *Let B be a sublattice of a finite product space.*

(1) The Greedy Algorithm returns Status = Feasible *and outputs a feasible solution x if and only if problem (P) is feasible.*

(2) If problem (P) is feasible, then the Greedy Algorithm outputs an optimal solution to problem (P) for all nonnegative demands d if and only if w is submodular.

(3) If problem (P) is feasible and w is submodular, then the Greedy Algorithm outputs an optimal solution to problem (D) in the Dual Phase.

Proof: First observe that the algorithm is finite and terminates after at most $\sum_{i=1}^{k} m(i)$ iterations. Next, for any vector $x \in \Re^B$, let $w \cdot x \doteq \sum_{a \in B^*}$ and, for any section $B(i,j)$, let $x(B(i,j)) \doteq \sum_{a \in B(i,j)} x_a$. Our proof uses the following claim.

Claim *Let $a, b \in B$ and let x be a feasible solution to problem (P) with $x_a > 0$ and $x_b > 0$. Define*

$$swap(x, a, b) \doteq x + \epsilon(e^{a \vee b} + e^{a \wedge b} - e^a - e^b),$$

where $\epsilon \doteq \min\{x_a, x_b\}$ and e^u is the unit vector in $\Re^B$ associated with each element $u \in B$ (that is, $e^u_v = 1$ if $v = u$, and 0 otherwise). Then $swap(x, a, b)$ is a feasible solution to problem (P) with $swap(x, a, b)_{a \vee b} > x_{a \vee b}$. Furthermore, if function $w : B \mapsto \Re$ is submodular, then $w \cdot swap(x, a, b) \leq w \cdot x$.

Proof: Under the assumptions of the claim, let $x' \doteq \text{swap}(x, a, b)$. By the definition of ϵ we have $x' \geq 0$. The only sections $B(i,j)$ where $x'(B(i,j))$ might differ from $x(B(i,j))$ are those where $a(i) \neq b(i)$ and $j \in \{a(i), b(i)\}$. However, we also have

$$(e^{a \vee b} + e^{a \wedge b} - e^a - e^b)(B(i,j)) = 0$$

for $j = a(i)$ and for $j = b(i)$. Hence $x'(B(i,j)) = x(B(i,j)) = d_{ij}$ for all i, j, implying that x' is a feasible solution to (P). We have $x'_{a \vee b} = x_{a \vee b} + \epsilon > x_{a \vee b}$. Finally, the inequality $w \cdot \text{swap}(x, a, b) \leq w \cdot x$ follows from the identity $w \cdot \xi^u = w(u)$ for all $u \in B$ and from the submodularity of w. The proof of the claim is complete. **2**

(1) Observe that the equalities $x(B(i,j)) + \delta_{ij} = d_{ij}$ for all i, j, and the nonnegativity conditions $x \geq 0$ and $\delta \geq 0$, are all maintained at every step of the algorithm. Also observe that if the algorithm returns *Status= Feasible*, then all $\delta_{ij} = 0$, and the output x is therefore a feasible solution to problem (P). Conversely, assume that problem (P) is feasible. Given the recursive nature of the Greedy algorithm, we only need to show that there exists a feasible solution x to problem (P) in which x_{a^1} equals $\xi \doteq \min\{d_{i,a(i)} : i \in K\}$, as prescribed by the Greedy algorithm. Indeed, since problem (P) is feasible, let x be a feasible solution to (P) with the largest possible value of x_{a^1}. If $x_{a^1} < \xi$, then for every $i \in K$ there exists $a^{(i)} \in B(i, a^1(i)) \setminus \{a^1\}$ with $x_{a^{(i)}} > 0$. Let $x^{[1]} \doteq x$ and $a^{[1]} \doteq a^{(1)}$. For $i = 2, \ldots, k$, let and $x^{[i]} \doteq \text{swap}(x^{[i-1]}, a^{[i-1]}, a^{(i)}$ and $a^{[i]} \doteq a^{[i-1]} \vee a^{(1)}$. Note that $a^{[k]} = a^1$ and, by repeated application of the above claim, $x^{[k]}$ is a feasible solution to (P) with $x^{[k]}_{a^1} > x_{a^1}$, a contradiction with the definition of x. Hence we must have $x_{a^1} = \xi$, and the proof of *(1)* is complete.

(2) First assume that w is submodular. We only need to show that there exists an optimal solution x to problem (P) in which x_{a^1} equals $\xi \doteq \min\{d_{i,a(i)} : i \in K\}$, as prescribed by the Greedy algorithm. Indeed, let x be an optimal solution to (P) with the largest possible value of x_{a^1}. As in the proof of *(1)* above, we show that if $x_{a^1} < \xi$, then there exists a feasible solution $x^{[k]}$ to (P) with $x^{[k]}_{a^1} > x_{a^1}$ and, using the last part of the claim, with $w \cdot x^{[k]} \leq w \cdot x$, a contradiction with the definition of x. Hence we must have $x_{a^1} = \xi$, proving that the Greedy algorithm correctly fixes the value of x_{a^1}. Conversely, assume that w is an arbitrary cost-function such that the Greedy algorithm constructs an optimal solution to problem (P) for all $d \geq 0$. For any two $a, b \in B$ distinct from their join and meet, we define a problem instance with $d_{ij} = 1$ if $j \in \{a(i), b(i)\}$, and 0 otherwise, for all $i \in K$. Thus, x' defined by $x'_u = 1$ if $u \in \{a, b\}$, and 0 otherwise, is a feasible solution to (P). However, the Greedy algorithm applied to this problem

instance produces an optimal solution x where $x_u = 1$ if $u \in \{a \vee b, a \wedge b\}$, and 0 otherwise. Therefore, we have

$$w(a \wedge b) + w(a \vee b) = w(x) \leq w(x') = w(a) + w(b).$$

Since this inequality holds for any $a, b \in B$, it shows that w is submodular.

(3) Assume that problem (P) is feasible and w is submodular. To simplify the proof below, we assume, without loss of generality, that $d_{ij} > 0$ for all i, j. Indeed, for every $i \in K$, we may delete every coordinate j such that $B(i,j) = \emptyset$ or $d_{ij} = 0$, setting $x_a \doteq 0$ for all $a \in B(i,j)$ and, as observed previously, setting y_{ij} to a sufficiently small (negative) value. Each chain A_i is then assumed to be renumbered consecutively after all these deletions. (Note however that we perform such deletions only once, before applying the Greedy algorithm. If $\operatorname{argmin}\{\delta_{\ell,a(\ell)} : \ell \in \Lambda\}$ contains more than a single element, then we arbitrarily choose one element i in it, and we keep all the other sections, even though some subsequent step will assign a zero value to the corresponding primal variable. This will be needed in order to construct a dual solution in the Dual Phase.)

Let y denote the dual solution constructed in the Dual Phase of the Greedy algorithm. First, observe that y satisfies the complementary slackness conditions since for each variable x_{a^t} selected in the Primal Phase, the corresponding constraint in problem (D) is satisfied with equality in the Dual Phase. Since the primal solution x is optimal, it suffices to show that y is a feasible solution to (D).

We use the Manhattan (or rectilinear) distance between lattice elements in B: for $a, b \in B$ let $d(a,b) = \sum_{i=1}^{k} |a(i) - b(i)|$. Note that $a \leq a' \leq a''$ implies $d(a,a') + d(a',a'') = d(a,a'')$. We also define the Manhattan (or L_1) norm $||a|| \doteq d(a,0) = \sum_{i=1}^{k} a_i$ for all $a \in A$.

Let the *path* $P \doteq (a^1, a^2, \ldots, a^T)$ denote the sequence of lattice elements found in the Primal Phase of the Greedy algorithm. Note that $a^t > a^{t+1}$ for all $t = 1, \ldots, T-1$.

For any $a \in B$, let the distance of a to P be $d(a,P) \doteq \min_t d(a,a^t)$. We shall prove by induction on $d(a,P)$ that y satisfies the constraints of (D) corresponding to every element $a \in B^*$. We have already observed that these constraints hold with equality for every a with $d(a,P) = 0$. So assume that all constraints of (D) corresponding to elements $a' \in B^*$ with $d(a',P) \leq r-1$ hold, and consider $a \in B^*$ with $d(a,P) = r \geq 1$. Since $a^T = \bigwedge B^* < a < \bigvee b^* = a^1$, let $a^u \doteq \bigvee\{a^t : a^t < a\}$ and $a^v \doteq \bigwedge\{a^t : a^t > a\}$. If $u = v+1$, then $a^v > a > a^{v+1}$ implies that a^{v+1} could not have been selected at step $v+1$ of the Primal phase. Hence $u > v+1$ and, letting $b \doteq a^{v+1}$, we have $a' \doteq a \wedge b < a < a'' \doteq a \vee b$. This implies that $d(a,a') \geq 1$ and $d(a,a'') \geq 1$. Now, for every $i \in K$, the definition of $a'(i)$ and $a''(i)$ implies that $|a(i) - a'(i)| + |a'(i) - b(i)| = |a(i) - b(i)| = |a(i) - a''(i)| + |a''(i) - b(i)|$, implying $d(a,a') + d(a',b) = d(a,b) = d(a,a'') + d(a'',b)$, and therefore $d(a',b) \leq r-1$ and $d(a'',b) \leq r-1$. Letting $y(a) \doteq \sum_{i:a(i)\geq 1} y_{i,a(i)}$, we have $y(a) = y(a') + y(a'') - y(b)$. Since $b \in P$, we have $y(b) = w(b)$. Since $d(a',P) \leq r-1$ and $d(a'',P) \leq r-1$ we have, by the inductive assumption, $y(a') \leq w(a')$ and $y(a'') \leq w(a'')$. Therefore:

$$\begin{aligned} y(a) &= y(a') + y(a'') - y(b) \\ &\leq w(a') + w(a'') - w(b) \\ &\leq w(a), \end{aligned}$$

where the last inequality follows from submodularity of w and the definition of a' and a''. This completes the proof that y is a feasible, and, therefore optimal, solution to the dual problem (D).

2

Note that, for a given demand vector d such that problem (P) is feasible, the primal solution constructed by the Greedy Algorithm does not depend on the cost function w, provided it is submodular. See [1] and [2] for a study of a similar property in the context of ordinary transportation and minimum cost network flow problems.

A consequence of Theorem 1 is that, when w is submodular, the inequalities of problem (D) form a *totally dual integral* (TDI) linear inequality system (see Edmonds and Giles [16], Hoffman [25], and Nemhauser and Wolsey [31] for a definition and properties of TDI systems).

4 Submodular costs and Monge properties.

The main purpose of this section is to point out and exploit the equivalence between submodularity of a function defined on a product of k chains, and the Monge condition of a k-dimensional array. We also introduce the concept of submodular sequences, and discuss its relationship with that of Monge sequences in two-dimensional arrays. In particular, we show that, for any *strictly* submodular two-dimensional array, the class of submodular sequences coincides with that of Monge sequences.

The concept of Monge sequences was introduced for two-dimensional arrays (matrices) by Hoffman in 1963 [24] in order to describe classes of transportation problems that are greedily solvable. A *Monge sequence* for a two-dimensional $n \times m$ array $C = (c[i,j])$ is a total ordering of the nm pairs (i,j) such that, whenever pair (i,j) precedes both pairs (i,ℓ) and (k,j),

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

The inequality in this definition has been independently observed and exploited in algorithms for a wide variety of problems, under various conditions about the indices i, j, k and ℓ. The most common condition is that $i < k$ and $j < \ell$: a two-dimensional array C satisfies the *Monge condition* if

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

holds for all i, j, k, ℓ with $i \leq k$ and $j \leq \ell$. Note that this definition is precisely that of the submodularity of the function defined by C on the product lattice $\{1,\ldots,n\} \times \{1,\ldots,m\}$. A square matrix satisfying the Monge condition is called a *distribution matrix* in [21].

When the only defined entries in the matrix are above the diagonal (thus forming a sublattice of the lattice A of matrix cells), this condition is also called the (concave) quadrangle inequality (e.g., [41]), and (as if to add to the confusion) functions defined by such an array are sometimes called concave functions (e.g., [29]). The computer science community has seen a flurry of activity on these and closely related concepts during the past few years. This activity was motivated in part by the seminal paper of Aggarwal et al. [4] on matrix searching, and also by a wide variety of applications to problems in such areas as computational geometry (e.g., [4], [3]), VLSI channel routing (e.g., [4], [5]), signal quantization [40], molecular biology (e.g., [35], [29]), dynamic lot sizing (the Wagner-Whitin problem) [7], flow shop scheduling [38], and the travelling salesman problem [32]. Because the field is now so vast, we have only given here a few indicative references. Further references can be found therein.

The concept of Monge arrays has recently been extended to k-dimensional arrays by Aggarwal and Park [5], [6]: a k-dimensional array C satisfies the *Monge condition* if every two-dimensional plane of C corresponding to fixed values of $k-2$ coordinates satisfies the Monge condition.

Actually, the Monge condition just defined is equivalent to the submodularity of the function $c : A \mapsto \mathcal{R}$ defined by the array C. Indeed, the following result (compare Proposition 2.4 in [5])

is a direct rephrasing of Theorems 3.1 and 3.2 in Topkis [37] for the case where A is a product of a finite number of chains, as is assumed throughout the present paper:

Theorem 4.1 *A function $c : A \mapsto \mathcal{R}$ is submodular if and only if it is submodular on every two-dimensional sublattice (plane) corresponding to fixed values of $k-2$ coordinates.*

As a consequence, many results on k-dimensional arrays satisfying the Monge condition (such as in Section 2 of [5] and in Section 2 in [11]) can be directly derived from known results on submodular functions. In addition, the deep theory of parametric lattice programming developed in Topkis [37] also applies to problems involving k-dimensional arrays satisfying the Monge condition. Conversely, the rich computer science literature on Monge and related arrays, in particular the fast algorithms developed for a variety of such problems, may also be exploited to study submodular functions on discrete product lattices. (A case in point is the dynamic lot-sizing problem considered in detail in Topkis, and for which fast algorithms were derived in [7] using the Monge condition.)

We now go back to Monge sequences. This concept was originally introduced by Hoffman for two-dimensional arrays, as seen above. It was further investigated, among others, in [36], [1], [2], and is closely related to the notion of *greedoids* (see [28]).

Some authors (e.g., Bein et al. [10], and Rudolf [34]) have recently investigated the possibility of extending this concept to higher dimensions. However, at present no approach seems to clearly dominate. Hence we restrict our discussion of relations between submodularity and the existence of Monge sequences to the two-dimensional case. The following notion arises naturally in the lattice framework: a *submodular sequence* for a two-dimensional $n \times m$ array $C = (c[i,j])$ is a total ordering of the nm pairs (i,j) such that, for any i, j, k and ℓ, at least one of the pairs $(i,j) \wedge (k,\ell)$ or $(i,j) \vee (k,\ell)$ precedes both pairs (i,j) and (k,ℓ).

Proposition 4.1 *A two-dimensional array is submodular (or, equivalently, satisfies the Monge condition) if and only if every submodular sequence is a Monge sequence.*

Proof: Consider any $(i,j) \in \{1,\ldots,n\} \times \{1,\ldots,m\}$, any $h \in \{1,\ldots,n\}$ and $k \in \{1,\ldots,m\}$. Note that if $i \leq h$ and $j \leq k$, or if $i \geq h$ and $j \geq k$, then $\{(i,k) \wedge (h,j), (i,k) \vee (h,j)\} = \{(i,j),(h,k)\}$ and thus (i,j) cannot precede both (i,k) and (h,j) in any submodular sequence. Hence, if the given array C satisfies the Monge condition, then every submodular sequence is Monge for C. Conversely, observe that in any submodular sequence at least one of the pairs (i,j) and $(i+1,j+1)$ precedes both $(i,j+1)$ and $(i+1,j$, for every i and j. Therefore, if all submodular sequences are Monge, then the inequality $c[i,j]+c[i+1,j+1] \leq c[i,j+1]+c[i+1,j]$ holds for all i and j, implying, by theorem 4.1, that the array C is submodular. **2**

It is not true that, in a submodular array, every Monge sequence is a submodular sequence. For example, consider a constant array C where all entries $c[i,j]$ have the same value. Then every sequence of the pairs (i,j) is a Monge sequence. However, the next result shows that Monge sequences coincide with submodular sequences when C satisfies a *strict* version of the Monge condition:

Proposition 4.2 *If a two-dimensional array C defines a* strictly *submodular function then a total ordering of the pairs (i,j) is a submodular sequence if and only if it is a Monge sequence.*

Proof: Let array C define a strictly submodular function. Assume that there exists a Monge sequence in which neither (i,j) nor (h,k) precedes both (i,k) and (h,j), for some $i \leq h$ and

$j \leq k$. Then at least one of the pairs (i,k) and (h,j) precedes both (i,j) and (h,k) in this Monge sequence. Hence, by the definition of Monge sequences, we have $c[i,k]+c[h,j] \leq c[i,j]+c[h,k]$, contradicting the strict Monge property for array C. Hence every Monge sequence must be submodular. The converse follows directly from Proposition 4.1. **2**

5 A class of problems with submodular costs.

We now consider a class of problems where all the elements of each chain A_i^* are resources of a certain type, located in some given space where distances are defined (a *metric space*). The lattice elements $a \in A$ correspond to sets, or *clusters*, of chain elements, one from each chain, and the cost of each cluster is determined by some function of the distances between the cluster elements. For example, in a physical distribution context, the resources may include customers sites, warehouses, trucks, truck depots, plants, etc. In a flexible manufacturing application (Crama and Spieksma [13]), the resources are the part types, the feeder slots, and the insertion points on the circuit board. We assume that each element of each chain is located at some point in a given metric space. (The points need not all be distinct.) The cost rate $w(a)$ of cluster a reflects various motions and shipment activities within the cluster, and is determined jointly by the pattern of such activities within the cluster, and by the distances between cluster elements. It is convenient to associate with each cluster $a \in A$ a complete undirected graph $G^a = (a, E)$, where a is identified with the set of its components, and with edge lengths d_{ij}^a equal to the distance between the points where the i-th and j-th cluster elements are located. Letting d^a denote the (symmetric) distance matrix (d_{ij}^a), the cost rate $w(a)$ for cluster a is then determined by the value $F(d^a)$ of a given function F. This function F, assumed to be identical for all clusters $a \in A$, may reflect a fixed transportation pattern, for example $F(d^a) = \sum_{i,j} r_{ij} d_{ij}^a$, where the coefficients r_{ij} represents a predetermined flow of activities, per unit of demand, between resource types i and j. A special case arises where all $r_{ij} = 1$, in which case $F(d^a) = K(d^a)$, the total cost of the *complete graph* G^a when the distances are d^a. On the other hand, the function F may reflect a local optimization of activities within cluster a. For example, $F(d^a)$ may be the total length of the *shortest spanning tree*, of the *shortest Hamiltonian path*, or of the *shortest Hamiltonian cycle* ("travelling salesman tour"), between all points in a cluster, reflecting various ways in which these activities may be conducted. Yet another example is the *diameter* of G^a, that is, $\mathrm{diam}(G^a) \doteq \max_{i,j} d_{ij}^a$, reflecting the minimum time to perform all the within-cluster activities simultaneously.

So, to each element j in each set A_i is associated an *abscissa* $c_{ij} \in \mathcal{R}$, with the usual distance

$$d_{ij}^a \doteq |c_{i,a(i)} - c_{j,a(j)}|.$$

First consider the case where the cost $w(a)$ of cluster $a \in A$ is its diameter, that is,

$$w(a) \doteq \max_i \; c_{i,a(i)} - \min_i \; c_{i,a(i)}.$$

Note that, when the points are on the line, this cost is equal to that given by the shortest spanning tree and the shortest Hamiltonian path, and is half that of the shortest Hamiltonian cycle through the points in the cluster. The following theorem shows that this function w is submodular.

Theorem 5.1 *If all the points are located on a line, then the function w defined by the diameter (the total length of the shortest spanning tree, shortest Hamiltonian path, shortest Hamiltonian cycle) within each cluster, is submodular.*

Proof: Assume all points are located on a line, with abcissae c_{ij} as described above. Observe that the meet and join of any two clusters $a, b \in A$ are as follows:

$(a \wedge b)(i) = a(i)$ if $c_{i,a(i)} \leq c_{i,b(i)}$,

$= b(i)$ otherwise, and

$(a \vee b)(i) = b(i)$ if $c_{i,a(i)} \leq c_{i,b(i)}$,

$= a(i)$ otherwise, for $i \in K$.

For $a \in A$, let i_a (resp., j_a)) be the index for which $c_{i_a,a(i_a)}$ is minimal (resp., maximal). In other words, i_a (j_a) corresponds to the leftmost (rightmost) element of cluster a. To simplify the notation let $c_{i_a} \doteq c_{i_a,a(i_a)}$ and assume, without loss of generality, that $c_{i_a} \leq c_{i_b}$. Hence $c_{i_{a\wedge b}} = c_{i_a}$.

Case I: $c_{j_a} \geq c_{j_b}$. It follows that $c_{j_{a\vee b}} = c_{j_a}$. Furthermore, the construction of $a \wedge b$ and $a \vee b$ implies that $c_{j_{a\wedge b}} \leq c_{j_b}$ and $c_{i_{a\vee b}} \geq c_{i_b}$. This yields:

$$\begin{aligned} w(a) + w(b) &= c_{j_a} - c_{i_a} + c_{j_b} - c_{i_b} \\ &\geq c_{j_{a\vee b}} - c_{i_{a\wedge b}} + c_{j_{a\wedge b}} - c_{i_{a\vee b}} \\ &= w(a \wedge b) + w(a \vee b). \end{aligned}$$

Case II: $c_{j_a} < c_{j_b}$. Hence $c_{j_{a\vee b}} = c_{j_b}$. We also have $c_{j_{a\wedge b}} \leq c_{j_a}$ and $c_{i_{a\vee b}} \geq c_{i_b}$. It follows that

$$\begin{aligned} w(a) + w(b) &= c_{j_a} - c_{i_a} + c_{j_b} - c_{i_b} \\ &\geq c_{j_{a\wedge b}} - c_{i_{a\wedge b}} + c_{j_{a\vee b}} - c_{i_{a\vee b}} \\ &= w(a \wedge b) + w(a \vee b). \end{aligned}$$

The proof is complete. **2**

This result also implies that cost functions defined by any fixed activity pattern (including the cost of the complete graph) are submodular:

Corollary 5.1 *If all the points are located on a line, then the function w defined by $w(a) \doteq \sum_{i,j} r_{ij} d^a_{ij}$ where the coefficients $r_{ij} \geq 0$ are independent of a, is submodular.*

Proof: As before, assume all points are located on a line, with abcissae c_{ij}. Fix i and j in $\{1, \ldots, k\}$ with $i \neq j$. First define the function $\gamma_{ij} : A_i \times A_j \mapsto \Re$ by $\gamma_{ij}(u, v) \doteq |c_{iu} - c_{jv}|$. The previous theorem implies that γ_{ij} is submodular on $A_i \times A_j$. Now define $w_{ij} : A \mapsto \Re$ by $w_{ij}(a) \doteq \gamma_{ij}(a(i), a(j))$. It follows immediately that w_{ij} is submodular on A. Finally, with all $r_{ij} \geq 0$, this in turn implies that $w = \sum_{ij} r_{ij} w_{ij}$ is also submodular on A. The proof is complete. **2**

When Theorem 5.1 or Corollary 5.1 applies, the corresponding problems (P) and (D) may be solved by the Greedy Algorithm of Section 3. This algorithm takes a particularly simple form for the k-index assignment problem: if the resources in each chain i are sorted from left to right on the line then, for $\ell = 1, \ldots, m$, the ℓ-th cluster in an optimal solution simply consist of the ℓ-th resource of each type.

References

[1] Adler, I., A.J. Hoffman, and R. Shamir (1990), "Monge and feasibility sequences in general flow problems," Rutcor Research Report 70–90, Rutgers University.

[2] Adler, I. and R. Shamir (1990), "Greedily solvable transportation networks and edge-guided vertex elimination," Rutcor Research Report 39–91, Rutgers University.

[3] Aggarwal A. and M.M. Klawe (1990), "Applications of generalized matrix searching to geometric algorithms," *Discrete Applied Mathematics* **27**, 3–23.

[4] Aggarwal A., M.M.Klawe, S. Moran, P. Shor, and R. Wilber (1987) "Geometric applications of a matrix-searching algorithm," *Algorithmica* **2**, 195–208.

[5] Aggarwal A. and J.K. Park (1989) "Sequential searching in multi-dimensional monotone arrays," Research Report RC 15128, IBM T.J. Watson Research Center, Yorktown Heights, NJ.

[6] Aggarwal A. and J.K. Park (1989) "Parallel searching in multi-dimensional monotone arrays," Research Report RC 14826, IBM T.J. Watson Research Center, Yorktown Heights, NJ.

[7] Aggarwal A. and J.K. Park (1992) "Improved algorithms for economic lot-size problems," to appear in *Operations Research.*

[8] Balas, E. and M.J. Saltzman (1989), "Facets of the three-index assignment polytope," *Discrete Applied Mathematics* **23**, 201–229.

[9] Bandelt, H.J., Y. Crama and F.C.R. Spieksma (1992), "Approximation algorithms for multidimensional assignment problems with decomposable costs," to appear in *Discrete Applied Mathematics.*

[10] Bein, W.W., P. Brucker and P.K. Pathak (1991), "Monge properties in higher dimensions," Preprint 134, Universität Osnabrück.

[11] Bein, W.W., P. Brucker, J.K. Park and P.K. Pathak (1992), "A Monge property for the d-dimensional transportation problem," to appear in *Discrete Applied Mathematics.*

[12] Birkhoff, G., *Lattice Theory,* (3rd ed.) American Mathematical Colloquium Publications, Vol 25, 1967.

[13] Crama, Y. and F.C.R. Spieksma (1992), "Approximation algorithms for three-dimensional assignment problems with triangle inequalities," *European Journal of Operational Research* **60**, 273–279.

[14] Edmonds, J. (1970), "Submodular functions, matroids and certain polyhedra," in R. Guy et al., edrs, *Combinatorial Structures and Their Applications,* Gordon and Breach, 69–87.

[15] Edmonds, J. (1971), "Matroids and the greedy algorithm," *Mathematical Programming* **1**, 127–136.

[16] Edmonds, J. and R. Giles (1974), "A min-max relation for submodular functions on graphs", *Annals of Discrete Mathematics* **1**, 185–204.

[17] Frank, A. and É. Tardos (1988), "Generalized polymatroids and submodular flows," *Mathematical Programming* **42**, 489–563.

[18] Frieze, A.M. and J. Yadegar (1981), "An algorithm for solving 3-dimensional assignment problems with applications to scheduling a teaching practice," *Journal of the Operational Research Society* **32**, 989–995.

[19] Fujishige, S., *Submodular Functions and Optimization*, North Holland, 1991.

[20] Fujishige, S. and N. Tomizawa (1983), "A note on submodular functions on distributive lattices," *Journal of the Operations Research Society of Japan* **26**, 309–318.

[21] Granot, F. and A.F. Veinott, Jr. (1985), "Substitutes, complements and ripples in network flows," *Mathematics of Operations Research* **10**, 471–497.

[22] Haley, K.B. (1962) "The solid transportation problem," *Operations Research* **10**, 448–463.

[23] Hoffman, A.J. (1963), "On simple linear programming problems," in V. Klee, edr., *Convexity: Proceedings of the Seventh Symposium in Pure Mathematics*, Proceedings of Symposia in Pure Mathematics, Vol. 7, American Mathematical Society, 317–327.

[24] Hoffman, A.J. (1974), "A generalization of max-flow min-cut," *Mathematical Programming* **6**, 352–359.

[25] Hoffman, A.J. (1985), "On greedy algorithms that succeed," in: I. Anderson, edr, *Surveys in Combinatorics*, Cambridge University Press, 97–112.

[26] Klawe, M.M. (1992), "Superlinear bounds for matrix searching problems," *J. of Algorithms* **13**, 55–78.

[27] Korte, B. and L. Lovász (1991), *Greedoids*. North Holland.

[28] Larmore, L.L. and B. Schieber (1991), "On-line dynamic programming with applications to the prediction of RNA secondary structure," *J. of Algorithms* **12**, 490–515.

[29] Lovász, L. (1983), "Submodular functions and convexity," in A. Bachem et al., edrs, *Mathematical Programming — The State of The Art*, Springer, 235–257.

[30] Nemhauser, G.L., and L.A. Wolsey (1988), *Integer and Combinatorial Optimization*, Wiley.

[31] Park, J.K. (1991), "A special case of the n-vertex traveling salesman problem that can be solved in $O(n)$ time," *Information Proc. Let.* **40**, 247–254.

[32] Pierskalla, W.P. (1968), "The multidimensional assignment problem," *Operations Research* **16**, 422–431.

[33] Ruediger, R. (1992), personal communication.

[34] Ruediger, R. (1992), "Recognition of d-dimensional Monge arrays," Report Nr. 230, Institute für Mathematik, Technische Universität Graz.

[35] Sankoff, D and J.B. Kruskal, edrs, (1983), *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley.

[36] Shamir, R. and B.L. Dietrich (1990), "Characterization and algorithms for greedily solvable transportation problems," *Proc. 1st ACM/SIAM Symposium on Discrete Algorithms*, 358–366.

[37] Topkis, D.M. (1978) "Minimizing a submodular function on a lattice," *Operations research* **26**, 305–321.

[38] Veinott, A.F., Jr. (1989), "Representation of general and polyhedral subsemilattices and sublattices of product spaces," *Linear Algebra and its Applications* **114/115**, 681–704.

[39] Wu, X. (1991) "Optimal quantization by matrix searching," *J. of Algorithms* **12**, 663–673.

[40] Yao, F.F. (1980) "Efficient dynamic programming using quadrangle inequalities," in *Proc. 12th ACM Symp. on Theory of Computing*, 429–435.

[41] Yemelichev, V.A., M.M. Kovalev, and M.K. Kratsov (1984), *Polytopes, Graphs and Optimisation.* Cambridge University Press.