Mathematics of Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Best Algorithms for Approximating the Maximum of a Submodular Set Function

G. L. Nemhauser, L. A. Wolsey,

To cite this article:
G. L. Nemhauser, L. A. Wolsey, (1978) Best Algorithms for Approximating the Maximum of a Submodular Set Function. Mathematics of Operations Research 3(3):177-188. http://dx.doi.org/10.1287/moor.3.3.177

MATHEMATICS OF OPERATIONS RESEARCH
Vol. 3, No. 3, August 1978
Printed in U.S.A.

BEST ALGORITHMS FOR APPROXIMATING THE MAXIMUM OF A SUBMODULAR SET FUNCTION*

G. L. NEMHAUSER† AND L. A. WOLSEY

Center for Operations Research & Econometrics
University of Louvain

A real-valued function z whose domain is all of the subsets of $N = \{1, \ldots, n\}$ is said to be submodular if $z(S) + z(T) \geqslant z(S \cup T) + z(S \cap T)$, $\forall S, T \subseteq N$, and nondecreasing if $z(S) \leqslant z(T)$, $\forall S \subset T \subseteq N$. We consider the problem $\max_{S \subset N} \{z(S) : |S| \leqslant K, z \text{ submodular and nondecreasing}, z(\emptyset) = 0\}$.

Many combinatorial optimization problems can be posed in this framework. For example, a well-known location problem and the maximization of certain boolean polynomials are in this class.

We present a family of algorithms that involve the partial enumeration of all sets of cardinality q and then a greedy selection of the remaining elements, $q = 0, \ldots, K-1$. For fixed K, the qth member of this family requires $O(n^{q+1})$ computations and is guaranteed to achieve at least

$$\left[1 - \left(\frac{K-q}{K}\right)\left(\frac{K-q-1}{K-q}\right)^{K-q}\right] \times 100 \text{ percent of the optimum value.}$$

Our main result is that this is the best performance guarantee that can be obtained by any algorithm whose number of computations does not exceed $O(n^{q+1})$.

1. Introduction. Let $N = \{1, \ldots, n\}$ be a finite set. A real-valued function z whose domain is all of the subsets of N is said to be *submodular* if

$$z(S) + z(T) \geqslant z(S \cup T) + z(S \cap T), \quad \forall S, T \subseteq N$$

and *nondecreasing* if $z(S) \leqslant z(T)$, $\forall S \subset T \subseteq N$.

Let $\mathfrak{Z}_n$ be the family of nondecreasing, submodular functions with $z(\emptyset) = 0$, $z \not\equiv 0$ and $|N| = n$; let $\mathfrak{Z} = \bigcup_{n=1}^{\infty} \mathfrak{Z}_n$. We assume that each member of the family $\mathfrak{Z}_n$ is described by a list of function values of length 2^n.

For each positive integer K we consider the family of problems

$$\max_{S \subseteq N} \{z(S) : |S| \leqslant K, z \in \mathfrak{Z}\}. \qquad (\mathfrak{P}_K)$$

A *problem instance* in the family $\mathfrak{P}_K$ is specified by a positive integer n and a $z \in \mathfrak{Z}_n$, and is denoted by $P_{n,z}$. The value of an optimal solution to $P_{n,z}$ is denoted by $Z(P_{n,z})$.

A subclass of $\mathfrak{Z}$ of practical interest can be generated from nonnegative matrices. Let $C = \{c_{ij}\}$, $i = 1, \ldots, m$, $j \in N = \{1, \ldots, n\}$ be a nonnegative matrix with the interpretation that c_{ij} is the value of assigning client i to location j. The set function $z(\emptyset) = 0$, $z(S) = \sum_{i=1}^{m} \max_{j \in S} c_{ij}$, $S \neq \emptyset$, is submodular and nondecreasing and $z(S)$ is the value of the subset of locations S. The problem of selecting an optimal set of locations S, subject to $|S| \leqslant K$, therefore belongs to $\mathfrak{P}_K$. It is a well-known *uncapacitated location problem* (see [1]).

* Received February 18, 1977; revised December 15, 1977.
AMS 1970 subject classification. Primary: 90C30.
IAOR 1973 subject classification. Main: Nonlinear programming.
Key words. Submodular functions, subadditive functions, computational complexity, best algorithms, polynomial running time, approximate optima, greedy algorithm.
† On leave of absence from Cornell University and supported, in part, by National Science Foundation Grant ENG75-00568.

This uncapacitated location problem is a special case of a matroid optimization problem that belongs to $\mathcal{P}_K$. This problem is to find a maximum weight independent set in a matroid when the elements of the matroid are colored and the elements of the independent set can have no more than K colors.

Another application involves the maximization of boolean polynomials. These and other classes of problems that motivated our work are described in [1] and [3].

An *algorithm* for $\mathcal{P}_K$ is simply a well-defined procedure for searching the list of function values that define z, together with a termination rule. An *approximation algorithm* is an algorithm that does not always find an optimal solution. Let $Z^A(P_{n,z})$ denote the minimum value[1] of a solution to $P_{n,z}$ produced by algorithm A. The *performance measure* of algorithm A over $\mathcal{P}_K$ is $J_K^A = \inf_{n,z\in\mathcal{Z}_n} Z^A(P_{n,z})/Z(P_{n,z})$. Note that algorithm A is exact if and only if $J_K^A = 1$.

Let $f_K^A(P_{n,z})$ be the maximum number of values of z required by algorithm A on $P_{n,z}$ and let $f_K^A(n) = \sup_{z\in\mathcal{Z}_n} f_K^A(P_{n,z})$. We say that the number of function values required by algorithm A over $\mathcal{P}_K$ is of *order* n^q (written $O(n^q)$) if, for all but a finite number of values of n, $C_1 n^q \leqslant f_K^A(n) \leqslant C_2 n^q$, for some integer q, $0 < C_1 < C_2$.

Over all $O(n^q)$ algorithms, where q is fixed between 1 and K, we are interested in determining the best possible performance. In §2 we give lower bounds on this best possible performance for $q = 1, \ldots, K$ and $K = 1, 2, \ldots$. We describe a family of algorithms $G(q)$, $q = 0, 1, \ldots, K-1$, that we introduced in [3]. The qth member of this family is $O(n^{q+1})$ and has performance measure

$$J_K^{G(q)} = 1 - \left(\frac{K-q}{K}\right)\left(\frac{K-q-1}{K-1}\right)^{K-q}.$$

§3 gives a particular family of set functions and shows (the details are given in an Appendix) that this family belongs to $\mathcal{Z}$.

Our main results appear in §4. Using the family of §3, we show that $J_K^{G(q)}$ is an upper bound on the performance of all $O(n^{q+1})$ algorithms. As a consequence of this result and the result of [3] on the performance of $G(q)$ we can conclude that $\{G(q)\}_{q=0}^{K-1}$ is an optimal family of algorithms for $\mathcal{P}_K$.

Next we consider the family of problems $\mathcal{P} = \bigcup_{K=1}^{\infty} \mathcal{P}_K$. We show that for this family there does not exist an $O(n^q)$ algorithm for any integer q (i.e., an algorithm requiring a number of function values polynomial in n) that can achieve a performance measure greater than $(e-1)/e$, where e is the base of the natural logarithm.

We conclude with a discussion of the results.

2. The q-enumeration plus greedy algorithm $[G(q)]$. Let $0 \leqslant q \leqslant K-1$ be an integer and $\rho_j(S) = z(S \cup \{j\}) - z(S)$. The algorithm has two simple parts as follows.

q-enumeration. Produce a list of all subsets of N of cardinality q.

greedy. Do for all sets of the list.

initialization. Let S^q be the set chosen from the list, $N^q = N - S^q$ and $t = q+1$.

iteration t. Select $i(t) \in N^{t-1}$ for which $\rho_{i(t)}(S^{t-1}) = \max_{i\in N^{t-1}} \rho_i(S^{t-1})$ with ties settled arbitrarily. Set $\rho_{t-1} = \rho_{i(t)}(S^{t-1})$. If $\rho_{t-1} > 0$, set $S^t = S^{t-1} \cup \{i(t)\}$ and $N^t = N^{t-1} - \{i(t)\}$; otherwise set $S^t = S^{t-1}$ and $N^t = N^{t-1}$. If $t = K$ or if $\rho_{t-1} = 0$ stop with the set S^t; otherwise set $t \leftarrow t+1$ and continue.

solution. Output the best solution found in the $\binom{n}{q}$ passes through greedy.

The number of function values required by $G(q)$ is at most

$$I(q) = \binom{n}{q}\left[(n-q) + (n-q-1) + \cdots + (n-K+1)\right].$$

$I(q)$ is proportional to Kn^{q+1} and, for fixed K, is $O(n^{q+1})$.

[1] Minimum value is required in this definition when the algorithm can produce alternate solutions of different value.

Define

$$\alpha_K^q = 1 - \left(\frac{K-q}{K}\right)\left(\frac{K-q-1}{K-q}\right)^{K-q},$$

for $q = 0, 1, \ldots, K-1$ and $K = 1, 2, \ldots$. In [3] we proved (see Theorems 4.1 and 7.1):

THEOREM 2.1. *$J_K^{G(q)} \geqslant \alpha_K^q$, for $q = 0, 1, \ldots, K-1$ and $K = 1, 2, \ldots$.*

We also proved in [3] that $J_K^{G(0)} = \alpha_K^0$, for $K = 1, 2, \ldots$. Here we prove that the bound of Theorem 2.1 is tight for all q and K, and more significantly, that if the performance measure of any algorithm exceeds α_K^q then that algorithm requires more than $O(n^{q+1})$ function values.

3. A family of nondecreasing submodular set functions. We define a family of submodular set functions that provide worst-case examples over all algorithms. The (r, K)th subfamily, $r = 1, \ldots, K-1$, $K = 2, 3, \ldots$, is specified by the functions $v_r^K \in \mathfrak{S}_n$, $n \geqslant 3(K-r) + r - 2$. (It is to be understood that v_r^K is a set function on a set N of cardinality n. For notational simplicity we have suppressed the dependence on n, although there is a different function for each integer $n \geqslant 3(K-r) + r - 2$.) We will show that $Z^A(P_{n, v_r^K})/Z(P_{n, v_r^K}) \leqslant \alpha_K^{r-1}$ for any algorithm A requiring no more than $O(n^r)$ function values.

The set $N = \{1, \ldots, n\}$ contains two types of elements called *special* and *plain*. The subset M, $|M| = K$, is the set of the special elements and $N - M$ is the set of plain elements. The value of $v_r^K(S)$, $S \subseteq N$, depends only on $|S|$, $|S \cap M|$, r and K. For

$K = 2$

j \ i	0	1	2
0	0		
1	2	2	
2	3	3	4
3	4	4	4
4	4	4	4
⋮	⋮		⋮
n	4	4	4

$K = 3$

j \ i	0	1	2	3
0	0			
1	9	9		
2	15	15	18	
3	19	19	21	27
4	23	23	23	27
5	25	25	25	27
6	27	27	27	27
7	27	27	27	27
⋮	⋮			⋮
n	27	27	27	27

$K = 4$

j \ i	0	1	2	3	4
0	0				
1	64	64			
2	112	112	128		
3	148	148	160	192	
4	175	175	184	208	256
5	202	202	202	220	256
6	229	229	220	229	256
7	238	238	238	238	256
8	247	247	247	247	256
9	256	256	256	256	256
10	256	256	256	256	256
⋮	⋮				⋮
n	256	256	256	256	256

FIGURE 1. Tables of $v_1^K(i, j)$, $K = 2, 3, 4$.

$r = 2$

j \ i	0	1	2	3	4
0	0				
1	9	9			
2	18	18	18		
3	24	24	24	27	
4	28	28	28	30	36
5	32	32	32	32	36
6	34	34	34	34	36
7	36	36	36	36	36
8	36	36	36	36	36
⋮	⋮				⋮
n	36	36	36	36	36

$r = 3$

j \ i	0	1	2	3	4
0	0				
1	2	2			
2	4	4	4		
3	6	6	6	6	
4	7	7	7	7	8
5	8	8	8	8	8
6	8	8	8	8	8
⋮	⋮				⋮
n	8	8	8	8	8

FIGURE 2. Tables of $v_2^4(i, j)$ and $v_3^4(i, j)$

this reason we write $v_r^K(S) \equiv v_r^K(|S \cap M|, |S|) = v_r^K(i, j)$, where $i = |S \cap M|$ and $j = |S|$, $i = 0, \ldots, \min(j, K)$, $j = 0, \ldots, n$.

The remainder of this section gives the formulas that define v_r^K. Much of the detail, however, is not important for obtaining the results. The significant properties of v_r^K, which can be seen in the examples of Figures 1 and 2, are

PROPERTY 1: v_r^K is submodular and nondecreasing for all r and K.

PROPERTY 2: For $i \leqslant r$ and all j, $v_r^K(i, j)$ does not depend on i or, in other words, $v_r^K(i, j) = v_r^K(0, j)$.

PROPERTY 3: $\max_{(i,j)} \{v_r^K(i, j) : j \leqslant K\} = v_r^K(K, K) = K(K - r + 1)^{K-r}$.

PROPERTY 4: Let $p_r^K = 3(K - r) + r - 2$. For all $j > p_r^K$ and all i, $v_r^K(i, j) = K(K - r + 1)^{K-r}$.

PROPERTY 5: $v_r^K(0, K)/v_r^K(K, K) = \alpha_K^{r-1}$.

Property 2 implies that function values do not allow us to distinguish among sets of a given cardinality containing r or fewer special elements. Property 4 implies that function values do not allow us to distinguish among sets of cardinality larger than p_r^K.

The family $\{v_1^K\}$ is defined as follows. For $K = 2, 3 \ldots$

$$v_1^K(0, 0) = 0; \tag{3.1}$$

$$v_1^K(0, j) = v_1^K(1, j), \qquad 1 \leqslant j \leqslant n; \tag{3.2}$$

$$v_1^K(i, j) = K^K - (K - 1)^{j-i} K^{K-j+i-1}(K - i), \quad i \leqslant j \leqslant K + i - 1,\ 1 \leqslant i \leqslant K; \tag{3.3}$$

$$v_1^K(i,j) = K^K - (K-1)^{K-2}[(K-i)(2K-2+i-j)],$$
$$K+i \leqslant j \leqslant 2K+i-3,\ 1 \leqslant i \leqslant K; \tag{3.4}$$

$$v_1^K(i,j) = K^K - (K-1)^{K-2}[3K-j-3],$$
$$2K+i-2 \leqslant j \leqslant 3K-3,\ 1 \leqslant i \leqslant K-1; \tag{3.5}$$

$$v_1^K(i,j) = K^K, \qquad 3K-2 \leqslant j \leqslant n,\ 1 \leqslant i \leqslant K. \tag{3.6}$$

Tables of $v_1^K(i,j)$ for $K = 2, 3, 4$ are given in Figure 1.

THEOREM 3.1. *The function v_1^K, $K = 2, 3, \ldots$, is submodular and nondecreasing for all $n \geqslant 3K-4$.*

This theorem is proved by a repeated and tedious application of a definition of nondecreasing, submodular functions and is deferred to an Appendix.

For $r \geqslant 2$, v_r^K is defined in terms of v_1^K as follows. For $r = 2, \ldots, K-1$ and $K = 3, 4, \ldots$,

$$v_r^K(0,0) = 0; \tag{3.7}$$

$$v_r^K(i,j) = v_r^K(j,j) = j(K-r+1)^{K-r}, \qquad 0 \leqslant i \leqslant j,\ 1 \leqslant j < r; \tag{3.8}$$

$$v_r^K(i,j) = v_r^K(r,j), \qquad 0 \leqslant i < r,\ r \leqslant j \leqslant n; \tag{3.9}$$

$$v_r^K(i,j) = (r-1)(K-r+1)^{K-r} + v_1^{K-r+1}(i-r+1, j-r+1),$$
$$r \leqslant i \leqslant j,\ r \leqslant j \leqslant n. \tag{3.10}$$

Tables of $v_r^4(i,j)$ for $r = 2, 3$ are given in Figure 2.

THEOREM 3.2. *The function v_r^K, $r = 2, \ldots, K-1$, $K = 3, 4, \ldots$ is submodular and nondecreasing for all $n \geqslant 3(K-r)+r-2$.*

The proof is given in the Appendix.

Theorems 3.1 and 3.2 are property 1 given above. Property 2 is a direct consequence of (3.2), (3.8) and (3.9). For $r = 1$ property 3 follows from (3.3), and then from $v_1^K(K,K) = K^K$, $v_1^{K-r+1}(K-r+1, K-r+1) = (K-r+1)^{K-r+1}$ and (3.10) we obtain property 3 for $r > 1$. For $r = 1$ property 4 follows from (3.5) and (3.6). This result and (3.10) imply property 4 for $r > 1$. To establish property 5 we note that from (3.2), (3.3), (3.9) and (3.10),

$$v_r^K(0,K) = (r-1)(K-r+1)^{K-r} + (K-r+1)^{K-r+1} - (K-r)^{K-r}(K-r)$$
$$= K(K-r+1)^{K-r} - (K-r)^{K-r+1}.$$

Thus

$$\frac{v_r^K(0,K)}{v_r^K(K,K)} = 1 - \frac{(K-r)^{K-r+1}}{K(K-r+1)^{K-r}} = 1 - \left(\frac{K-r+1}{K}\right)\left(\frac{K-r}{K-r+1}\right)^{K-r+1} = \alpha_K^{r-1}.$$

4. Number of function values to exceed specified performance measure. In this section we specify the desired performance measure and calculate the number of function values required by any algorithm whose performance strictly exceeds this threshold level. To be precise we examine algorithms for $\mathcal{P}_K$ for which $J_K^A > \alpha_K^{r-1}$, or in other words algorithms for the problem:

Find $S \subseteq N$, $|S| \leqslant K$ so that $z(S)/Z(P_{n,z}) > \alpha_K^{r-1}$, $z \in \mathcal{Z}$.

Throughout the section, problems often will be specified with information additional to $z \in \mathfrak{Z}$ given in the problem statement. Clearly, as the information increases, the problem does not become more difficult.

We consider a very special case of the above problem:

$$\text{Find } S \subseteq N \text{ with } |S| \leqslant K \text{ so that } \frac{v_r^K(S)}{K(K-r+1)^{K-r}} > \alpha_K^{r-1}, \tag{4.1}$$

where it is specified in the problem statement that $z \equiv v_r^K$, the function given in §3, but M, the set of special elements, is unknown.

Note that by property 3 of §3, S is a solution to (4.1) if and only if $|S| \leqslant K$ and $z(S)/Z(P_{n,z}) > \alpha_K^{r-1}$, where $z \equiv v_r^K$.

For given r and K we will find a lower bound on the number of function values required to solve (4.1). We proceed in two steps. We first specify a combinatorial problem with the property that the minimum number of function values required to solve this problem by any algorithm is less than or equal to that for any algorithm for (4.1). Then we calculate a lower bound on the number of function values required to solve the combinatorial problem by any algorithm.

Consider the set N, defined in the previous section, containing a set M of special elements, and the problem:

> Find a set $S \subseteq N$ with $|S| \leqslant p_r^K$, $|S \cap M| \geqslant r+1$, where M is unknown, and if a set S is proposed, we are informed whether S is a solution of the problem or not. (4.2)

We assume that a proposal and a reply counts as one function value. Note that both problems (4.1) and (4.2) are such that any algorithm can terminate when a set S that is a solution to the problem is proposed, i.e., we can assume that no set examined before the last one is a solution.

LEMMA 4.1. *The minimum number of function values required to solve* (4.2) *is a lower bound on the number required to solve* (4.1).

PROOF. Consider the problem:

> Find a set $S \subseteq N$ with $|S| \leqslant p_r^K$, $|S \cap M| \geqslant r+1$, where it is specified in the problem statement that $z \equiv v_r^K$ but M is unknown, and if a set S is proposed we are given (i) the value of $v_r^K(S)$, and (ii) we are informed whether or not S is a solution. (4.3)

We will prove the lemma by showing that (a) the number of function values required to solve (4.3) is equal to or less than the number required to solve (4.1) and then (b) that the information contained in (4.3) that is not contained in (4.2) is worthless. Thus (4.2) requires the same number of function values as (4.3), which is equal to or less than the number required by (4.1).

Note that by properties 1 (nondecreasing), 2 and 5 of v_r^K, any solution to (4.1) has $|S| \leqslant K \leqslant p_r^K$ and $|S \cap M| \geqslant r+1$; hence it is a solution to (4.3). Thus (a) is proved.

Comparing problems (4.3) and (4.2) we see that they are identical except that more information is available in (4.3), and hence this problem may be solvable with fewer function values. To prove (b) we show that this additional information is worthless. If $S = (i, j)$ is not a solution of (4.2) or (4.3), then $v_r^K(i, j) = v_r^K(0, j)$. Therefore as long as an algorithm observes sets S that are not solutions, the information obtained in solving (4.2) or (4.3) is identical. We learn only that $S = (i, j)$ is not a solution of the problem. However as soon as a solution S is encountered, both algorithms stop. ∎

We now derive a lower bound on the minimum number of function values required by any algorithm to solve the (r, K) member of (4.2).

LEMMA 4.2. *The nnmber of function values required by any algorithm to solve the* (r, K) *member of* (4.2) *is at least*

$$\theta_r^K = \binom{n}{K} \Big/ \binom{p_r^K}{r+1} \binom{n-r-1}{K-r-1}.$$

PROOF. Let $\{Q_t\}_{t \in \mathfrak{T}}$ be the set of all subsets of N of cardinality p_r^K. As S is not a solution whenever $|S| > p_r^K$, and $S_1 \subset S_2 \Rightarrow |S_1 \cap M| \leqslant |S_2 \cap M|$, we can restrict our attention to algorithms that consider only Q_t, $t \in \mathfrak{T}$. Now we show that any algorithm that chooses the sets $\{Q_t\}_{t \in T}$, $T \subset \mathfrak{T}$ in any order cannot be guaranteed to solve (4.2) unless $|T| \geqslant \theta_r^K$.

Consider some set Q_t, $t \in \mathfrak{T}$. Let n^* be the number of K-tuples of N that have at least $(r+1)$ elements in common with Q_t. Then $n^*|T|$ is an upper bound on the number of K-tuples having at least $(r+1)$ elements in common with one of the sets $\{Q_t\}_{t \in T}$. Suppose $n^*|T| < \binom{n}{K}$, the total number of K-tuples of N. Then at least one K-tuple, say R, does not have more than r elements in common with any of the sets $\{Q_t\}_{t \in T}$. If $R \equiv M$, the set of special elements, then the algorithm observing $\{Q_t\}_{t \in T}$ cannot have solved (4.2) since $|Q_t \cap M| \leqslant r$ for all $t \in T$. Thus $|T| \geqslant \binom{n}{K}/n^*$. Finally we show that

$$n^* \leqslant \binom{n-r-1}{K-r-1} \binom{p_r^K}{r+1}.$$

Let $\{i_1, \ldots, i_{r+1}\} \subseteq R_s \cap Q_t$ for some K-tuple R_s. The number of K-tuples that contain the set $\{i_1, \ldots, i_{r+1}\}$ is $\binom{n-r-1}{K-r-1}$. The number of distinct $(r+1)$-tuples in Q_t is $\binom{p_r^K}{r+1}$ and the result follows. ∎

We now have all that we need for the main result.

THEOREM 4.1. *For the family of problems* $\mathcal{P}_K$ *no* $O(n^{q+1})$ *algorithm has a performance measure exceeding* α_K^q. *The algorithm* $G(q)$ *is* $O(n^{q+1})$ *and* $J_K^{G(q)} = \alpha_K^q$.

PROOF. From Lemmas 4.1 and 4.2 any algorithm that has a performance measure exceeding α_K^q requires at least θ_{q+1}^K function values. But K is fixed, $q < K$ and p_q^K is a linear function of q and K, which implies that θ_{q+1}^K is proportional to $n!/(n-q-2)!$ and therefore is $O(n^{q+2})$. The results about $G(q)$ now follow from Theorem 2.1. ∎

On the basis of this result it is reasonable to call $G(q)$ an optimal $O(n^{q+1})$ algorithm for $\mathcal{P}_K$ and to say that the family $\{G(q)\}_{q=0}^{K-1}$ contains optimal algorithms of all integral orders. However our result does not preclude the possibility of $O(n^{q+1})$ algorithms that achieve the performance measure α_K^q but require fewer function values than $G(q)$; nor does it rule out better algorithms than $G(q)$ for small values of n.

The results we have just given for fixed K can be generalized to allow K to be part of the problem statement.

Consider the family of problems

$$\max_{S \subseteq N} \{z(S) : |S| \leqslant \lambda, z \in \mathfrak{Z}\}. \qquad (\mathcal{P})$$

A problem instance in the family $\mathcal{P}$, denoted by $P_{n,z,\lambda}$, is specified by a positive integer n, an integer λ, $1 \leqslant \lambda \leqslant n$, and a $z \in \mathfrak{Z}_n$. The performance measure of algorithm A over $\mathcal{P}$ is $J^A = \inf_{n,z,\lambda} Z^A(P_{n,z,\lambda})/Z(P_{n,z,\lambda})$, where Z^A and Z are as defined in §1. The number of function values required by algorithm A over $\mathcal{P}$ is $O(n^q)$ if $f^A(n) = \sup_{z \in \mathfrak{Z}_n, \lambda} f^A(P_{n,z,\lambda})$ is $O(n^q)$, where $f^A(P_{n,z,\lambda})$ is the maximum number of function values required by A on $P_{n,z,\lambda}$.

THEOREM 4.2. *For the family of problems $\mathcal{P}$ no algorithm requiring a number of function values polynomial in n has a performance measure that exceeds $(e-1)/e$. The greedy algorithm is $O(n^2)$ and has performance measure $(e-1)/e$.*

PROOF. The second statement of the theorem follows from the results of §2 with $q=0$ and

$$J^{G(0)} = \inf_K \left[1 - \left(\frac{K-1}{K} \right)^K \right] = (e-1)/e.$$

Now consider a fixed integer q and a β, $(e-1)/e < \beta \leqslant 1$. We will show that for any β and q there exists a problem family $\{P_{n, z_n, K(n)}\}_{n=n^*}^{\infty} \subset \mathcal{P}$ for which no $O(n^{q+1})$ algorithm can achieve a performance measure of β. The family is defined as follows.

Take any sequence $\{K(n)\}_{n=1}^{\infty}$ where $1 \leqslant K(n) \leqslant n$, $K(n) \to \infty$ as $n \to \infty$ and $K(n)$ is bounded from above by an $O(n^{\gamma})$ polynomial, $\gamma < \frac{1}{2}$. Noting that

$$\alpha_{K(n)}^q = 1 - \left(\frac{K(n)-q}{K(n)} \right) \left(\frac{K(n)-q-1}{K(n)-q} \right)^{K(n)-q}$$

decreases with n, and has limit $(e-1)/e$, let n^* be the smallest integer for which $\alpha_{K(n^*)}^q < \beta$. Finally let $z_n = v_q^{K(n)}$.

By Lemmas 4.1 and 4.2 any algorithm for $P_{n, z_n, K(n)}$, $n \geqslant n^*$, that has a performance measure exceeding $\alpha_{K(n)}^q$ requires at least $\theta_{q+1}^{K(n)}$ function values. Since q is fixed, $p_q^{K(n)}$ is a linear function of $K(n)$, and hence $\theta_{q+1}^{K(n)}$ is $O(n^{q+2}/K(n)^{2q+4})$. Since $K(n)$ is bounded from above by an $O(n^{\gamma})$ polynomial, $\theta_{q+1}^{K(n)}$ is bounded from below by an $O(n^{(q+2)(1-2\gamma)})$ polynomial. For $\gamma < \frac{1}{2}$ such a polynomial is $O(n^{c(q+2)})$ where c is a positive constant. Now the theorem follows because q can be any positive integer, and β can be arbitrarily close to $(e-1)/e$. ∎

Note that the proof requires $K(n)$ to be bounded from above by a $O(n^{\gamma})$ polynomial, $\gamma < \frac{1}{2}$. We do not have results, for example, for $K(n)$ linear in n, although we do know [1] that the greedy algorithm can achieve its worst case performance for a family of problems with $K = n/2$.

5. Discussion. The results of §4 depend upon the assumption that each member of $\mathcal{Z}$ is encoded by a list of function values. Any complete description of $\mathcal{Z}$ requires an encoding by a list whose size grows exponentially with n. Although function values seem to be the most natural representation, members of $\mathcal{Z}$ can be described by different information. For example, in [3] we have given a representation of all members of $\mathcal{Z}$ in terms of coefficients of boolean polynomials. For this representation we would guess that results similar to the ones given here could be obtained.

It would be interesting and perhaps practically significant to obtain bounds on the performance of all algorithms with a specified amount of computation for subclasses of $\mathcal{P}_K$ generated from subclasses of $\mathcal{Z}$; one example is the *uncapacitated location problem* described in the introduction.

It is shown in [1] that the greedy algorithm has the same performance for the uncapacitated location problem as it does for $\mathcal{P}_K$. Furthermore, the greedy algorithm and most other approximation algorithms in the literature surprisingly use the matrix C only to calculate function values. Such algorithms treat the uncapacitated location problem as if it were an unstructured subclass of $\mathcal{P}_K$. We call them "black-box" algorithms, since we imagine that they use a black-box or subroutine that contains the "natural" problem description but gives only function values.

A natural extension of the results of this paper would be to analyze the performance of black-box algorithms for the uncapacitated location problem and other

subclasses of $\mathcal{P}_K$. In a forthcoming paper we will show that black-box algorithms, whose number of computations are polynomial in n, cannot solve the uncapacitated location problem. Still many interesting questions remain to be answered. What are the limitations of black-box or function value algorithms for subclasses of $\mathcal{P}_K$? With an amount of computation polynomial in n, can we achieve better performance by using the problem data to get information other than function values?

Finally, we would like to place our work within the framework of standard models in the theory of computational complexity by obtaining performance bounds for subclasses of $\mathcal{P}_K$ in which the size of the natural description of the problem is polynomial in n. Note that Theorem 4.2 is not within this framework and therefore sheds no light on the existence of polynomial algorithms for integer programming and other difficult combinatorial problems since we have measured computation as a function of n rather than list size.

Appendix. Let $\rho_k(S) = z(S \cup \{k\}) - z(S)$. $z(S)$ is nondecreasing if and only if $\rho_k(S) \geqslant 0$ $\forall S \subset N$ and $k \in N - S$, and submodular if and only if $\rho_k(S) \geqslant \rho_k(T)$ $\forall S \subset T \subset N$ and $k \in N - T$ (see [3]).

With $S = (i, j)$ and $z(S) = v_r^K(S) = v_r^K(i, j)$ as defined in §3

$$v_r^K(S \cup \{k\}) = \begin{cases} v_r^K(i+1, j+1) & \text{if } k \in M, \\ v_r^K(i, j+1) & \text{if } k \in N - M. \end{cases}$$

To simplify notation in the proof of Theorem 3.1 we let $v(i, j) = v_1^K(i, j)$ and

$$\rho_k(i, j) = \begin{cases} v(i+1, j+1) - v(i, j) & \text{if } k \in M, \\ v(i, j+1) - v(i, j) & \text{if } k \in N - M. \end{cases}$$

We will prove Theorem 3.1 by showing that $\rho_k(i, j) \geqslant 0$ (nondecreasing) and

$$\rho_k(i, j) \leqslant \min\{\rho_k(i-1, j-1), \rho_k(i, j-1)\}$$

(submodularity) for the function $v(i, j)$ defined by (3.1)–(3.6). The proof is straightforward but involves many cases.

PROOF OF THEOREM 3.1. I. $v(i, j)$ is nondecreasing, $\rho_k(i, j) \geqslant 0$.

Case A. $k \in M$, $\rho_k(i, j) = v(i+1, j+1) - v(i, j)$.

(a) $i = j = 0$.

$$\rho_k(0, 0) = v(1, 1) - 0 = K^K - K^{K-1}(K-1) = K^{K-1}.$$

(b) $i = 0, j > 0$.

$$\rho_k(0, j) = v(1, j+1) - v(0, j) = v(1, j+1) - v(1, j)$$

from (3.2), and hence $\rho_k(0, j)$ is the same for all $k \in N$. We will treat this case under plain elements.

(c) $1 \leqslant i \leqslant K-1$, $i \leqslant j \leqslant K + i - 1$. $v(i, j)$ and $v(i+1, j+1)$ are defined by (3.3).

$$\begin{aligned} \rho_k(i, j) &= (K-1)^{j-i} K^{K-j+i-1}\big[(K-i) - (K-i-1)\big] \\ &= (K-1)^{j-i} K^{K-j+i-1} \geqslant 0. \end{aligned}$$

(d) $1 \leqslant i \leqslant K-1$, $K + i \leqslant j \leqslant 2K + i - 3$. $v(i, j)$ and $v(i+1, j+1)$ are defined by (3.4).

$$\begin{aligned} \rho_k(i, j) &= (K-1)^{K-2}(2K - 2 + i - j)\big[(K-i) - (K-i-1)\big] \\ &= (K-1)^{K-2}(2K - 2 + i - j) \geqslant 0. \end{aligned}$$

(e) $1 \leqslant i \leqslant K-2$, $2K+i-2 \leqslant j \leqslant 3K-4$. $v(i,j)$ and $v(i+1,j+1)$ are defined by (3.5).

$$\rho_k(i,j) = (K-1)^{K-2} \geqslant 0.$$

(f) $1 \leqslant i \leqslant K-1$, $j \geqslant 3K-3$. $v(i,j) = K^K$, $\rho_k(i,j) = 0$.

Case B. $k \in N - M$, $\rho_k(i,j) = v(i,j+1) - v(i,j)$.

(a) $i = j = 0$.

$$\rho_k(0,0) = v(0,1) - v(0,0) = v(1,1) - v(0,0) = K^{K-1}.$$

(b) $i = 0$, $j > 0$.

$$\rho_k(0,j) = v(0,j+1) - v(0,j) = v(1,j+1) - v(1,j) = \rho_k(1,j).$$

Covered in cases below under $i = 1$.

(c) $1 \leqslant i \leqslant K$, $i \leqslant j < K+i-1$. $v(i,j)$ and $v(i,j+1)$ are defined by (3.3).

$$\begin{aligned}\rho_k(i,j) &= (K-i)(K-1)^{j-i}K^{K-j+i-2}\big[K-(K-1)\big] \\ &= (K-i)(K-1)^{j-i}K^{K-j+i-2} \geqslant 0.\end{aligned}$$

(d) $1 \leqslant i \leqslant K$, $j = K+i-1$. $v(i,j)$ and $v(i,j+1)$ are defined by (3.3) and (3.4), respectively.

$$\begin{aligned}\rho_k(i,j) &= (K-i)\big[(K-1)^{K-1} - (K-1)^{K-2}(K-2)\big] \\ &= (K-i)(K-1)^{K-2} \geqslant 0.\end{aligned}$$

(e) $1 \leqslant i \leqslant K$, $K+i \leqslant j < 2K+i-3$. $v(i,j)$ and $v(i,j+1)$ are defined by (3.4).

$$\begin{aligned}\rho_k(i,j) &= (K-1)^{K-2}(K-i)\big[(2K-2+i-j) - (2K-2+i-j-1)\big] \\ &= (K-1)^{K-2}(K-i) \geqslant 0.\end{aligned}$$

(f) $1 \leqslant i \leqslant K-1$, $j = 2K+i-3$. $v(i,j)$ and $v(i,j+1)$ are defined by (3.4) and (3.5), respectively.

$$\rho_k(i,j) = (K-1)^{K-2}\big[(K-i) - (K-i-1)\big] = (K-1)^{K-2} \geqslant 0.$$

(g) $1 \leqslant i \leqslant K-2$, $2K+i-2 \leqslant j \leqslant 3K-4$. $v(i,j)$ and $v(i,j+1)$ are defined by (3.5).

$$\rho_k(i,j) = (K-1)^{K-2} \geqslant 0.$$

(h) $1 \leqslant i \leqslant K$, $j \geqslant 3K-3$. $v(i,j) = K^K$, $\rho_k(i,j) = 0$.

II. $v(i,j)$ is submodular, $\rho_k(i,j) \leqslant \min\{\rho_k(i-1,j-1), \rho_k(i,j-1)\}$.

Case A. $k \in M$.

Case A1. $\rho_k(i,j) \leqslant \rho_k(i-1,j-1)$.

(a) $i = j = 1$. $\rho_k(1,1) = \rho_k(0,0) = K^{K-1}$.

(b) $i = 1$, $2 \leqslant j \leqslant K$. $\rho_k(1,j)$ is defined in I.A.(c) and $\rho_k(0,j-1)$ is defined in I.B.(c) with $i = 1$.

$$\rho_k(1,j) = (K-1)^{j-1}K^{K-j} = (K-1)(K-1)^{j-2}K^{K-j} = \rho_k(0,j-1).$$

(c) $i = 1$, $j = K+1$. $\rho_k(1,j)$ is defined in I.A.(d) and $\rho_k(0,j-1)$ is defined in I.B.(d) with $i = 1$.

$$\rho_k(1,j) = (K-1)^{K-2}(K-2) \leqslant (K-1)^{K-2}(K-1) = \rho_k(0,j-1).$$

(d) $i = 1$, $K+1 < j \leqslant 2K-2$. $\rho_k(1,j)$ is defined in I.A.(d) and $\rho_k(0,j-1)$ is defined in I.B.(e) with $i = 1$.

$$\rho_k(1,j) = (K-1)^{K-2}(2K-1-j) \leqslant (K-1)^{K-2}(K-1) = \rho_k(0,j-1).$$

(e) $i = 1$, $2K - 1 \leqslant j \leqslant 3K - 4$. $\rho_k(1, j)$ is defined in I.A.(e) and $\rho_k(0, j-1)$ is defined in I.B.(f) or I.B.(g) with $i = 1$. $\rho_k(1, j) = (K-1)^{K-2} = \rho_k(0, j-1)$.

(f) $i = 1$, $j \geqslant 3K - 3$. $\rho_k(1, j)$ is defined in I.A.(f). $\rho_k(1, j) = 0$.

(g) $1 < i \leqslant K$, $i \leqslant j \leqslant 3K - 4$. Both $\rho_k(i, j)$ and $\rho_k(i-1, j-1)$ are defined by I.A.(c) or by I.A.(d) or by I.A.(e). In each of these cases $\rho_k(i, j)$ is a function of $i - j$ or independent of i and j. Thus $\rho_k(i, j) = \rho_k(i-1, j-1)$.

(h) $1 < i \leqslant K$, $j \geqslant 3K - 3$. $\rho_k(i, j)$ is defined in I.A.(f). $\rho_k(i, j) = 0$.

Case A2. $\rho_k(i, j) \leqslant \rho_k(i, j-1)$.

(a) $i = 0$. $\rho_k(0, j)$ is the same for special and plain elements as indicated in I.A.(b). This case will be treated under case B2.

(b) $1 \leqslant i \leqslant K$, $i < j \leqslant K + i - 1$. $\rho_k(i, j)$ and $\rho_k(i, j-1)$ are defined in I.A.(c).

$$\rho_k(i, j) = (K-1)^{j-i} K^{K-j+i-1} \leqslant (K-1)^{j-i-1} K^{K-j+i} = \rho_k(i, j-1).$$

(c) $1 \leqslant i \leqslant K$, $j = K + i$. $\rho_k(i, j)$ and $\rho_k(i, j-1)$ are defined in I.A.(d) and I.A.(c), respectively.

$$\rho_k(i, j) = (K-1)^{K-2}(K-2) \leqslant (K-1)^{K-1} = \rho_k(i, j-1).$$

(d) $1 \leqslant i \leqslant K$, $K + i < j \leqslant 2K + i - 3$. $\rho_k(i, j)$ and $\rho_k(i, j-1)$ are defined in I.A.(d).

$$\rho_k(i, j) - \rho_k(i, j-1) = -(K-1)^{K-2} \leqslant 0.$$

(e) $1 \leqslant i \leqslant K - 2$, $j = 2K + i - 2$. $\rho_k(i, j)$ and $\rho_k(i, j-1)$ are defined in I.A.(e) and I.A.(d), respectively.

$$\rho_k(i, j) = (K-1)^{K-2} = \rho_k(i, j-1).$$

(f) $1 \leqslant i \leqslant K - 3$, $2K + i - 1 \leqslant j \leqslant 3K - 4$. $\rho_k(i, j)$ and $\rho_k(i, j-1)$ are defined in I.A.(e). $\rho_k(i, j) = \rho_k(i, j-1) = (K-1)^{K-2}$.

(g) $1 \leqslant i \leqslant K$, $j \geqslant 3K - 3$. $\rho_k(i, j)$ is defined in I.A.(f). $\rho_k(i, j) = 0$.

Case B. $k \in N - M$.

Case B1. $\rho_k(i, j) \leqslant \rho_k(i-1, j-1)$.

(a) $i = j = 1$. $\rho_k(1, 1)$ and $\rho_k(0, 0)$ are defined in I.B.(c) and I.B.(a), respectively.

$$\rho_k(1, 1) = (K-1)K^{K-2} \leqslant K^{K-1} = \rho_k(0, 0).$$

(b) $i = 1$, $j > 1$. $\rho_k(0, j-1) = \rho_k(1, j-1)$. This case will be covered under II.B2.

(c) $1 < i \leqslant K$, $i \leqslant j < K + i - 1$. $\rho_k(i, j)$ and $\rho_k(i-1, j-1)$ are defined in I.B.(c).

$$\begin{aligned}\rho_k(i, j) - \rho_k(i-1, j-1) &= (K-1)^{j-i} K^{K-j+i-2}[(K-i) - (K-i+1)] \\ &= -(K-1)^{j-i} K^{K-j+i-2} \leqslant 0.\end{aligned}$$

(d) $1 \leqslant i \leqslant K$, $K + i - 1 \leqslant j < 2K + i - 3$. Both $\rho_k(i, j)$ and $\rho_k(i-1, j-1)$ are defined in I.B.(d) or I.B.(e).

$$\begin{aligned}\rho_k(i, j) - \rho_k(i-1, j-1) &= (K-1)^{K-2}[(K-i) - (K-i+1)] \\ &= -(K-1)^{K-2} \leqslant 0.\end{aligned}$$

(e) $1 \leqslant i \leqslant K - 1$, $2K + i - 3 \leqslant j \leqslant 3K - 4$. Both $\rho_k(i, j)$ and $\rho_k(i-1, j-1)$ are defined in I.B.(f) or I.B.(g).

$$\rho_k(i, j) = \rho_k(i-1, j-1).$$

(f) $1 \leqslant i \leqslant K$, $j \geqslant 3K - 3$. $\rho_k(i, j)$ is defined in I.B.(h). $\rho_k(i, j) = 0$.

Case B2. $\rho_k(i, j) \leqslant \rho_k(i, j-1)$.

(a) $i = 0$. $\rho_k(0, j) = \rho_k(1, j)$. Covered in cases below under $i = 1$.

(b) $1 \leqslant i \leqslant K$, $i < j < K + i - 1$. $\rho_k(i, j)$ and $\rho_k(i, j-1)$ are defined in I.B.(c).

$$\rho_k(i,j) - \rho_k(i,j-1) = (K-i)(K-1)^{j-i-1}K^{K-j+i-2}\big[(K-1)-K\big]$$
$$= -(K-i)(K-1)^{j-i-1}K^{K-j+i-2} \leqslant 0.$$

(c) $1 \leqslant i \leqslant K$, $j = K + i - 1$. $\rho_k(i, j)$ and $\rho_k(i, j-1)$ are defined in I.B.(d) and I.B.(c), respectively.

$$\rho_k(i,j) - \rho_k(i,j-1) = (K-i)\big[(K-1)^{K-2} - (K-1)^{K-2}\big] = 0.$$

(d) $1 \leqslant i \leqslant K$, $K + i \leqslant j < 2K + i - 3$. $\rho_k(i, j)$ and $\rho_k(i, j-1)$ are defined by either I.B.(d) or I.B.(e).

$$\rho_k(i,j) = \rho_k(i,j-1).$$

(e) $1 \leqslant i \leqslant K - 1$, $j = 2K + i - 3$. $\rho_k(i, j)$ and $\rho_k(i, j-1)$ are defined in I.B.(f) and I.B.(e), respectively.

$$\rho_k(i,j) - \rho_k(i,j-1) = (K-1)^{K-2}\big[1-K+i\big] \leqslant 0.$$

(f) $1 \leqslant i \leqslant K - 2$, $2K + i - 2 \leqslant j \leqslant 3K - 4$. $\rho_k(i, j)$ and $\rho_k(i, j-1)$ are defined by either I.B.(f) or I.B.(g).

$$\rho_k(i,j) = \rho_k(i,j-1).$$

(g) $1 \leqslant i \leqslant K$, $j \geqslant 3K - 3$. $\rho_k(i, j)$ is defined by I.B.(h). $\rho_k(i, j) = 0$. ∎

PROOF OF THEOREM 3.2. Let

$$\rho_r^K(i,j,k) = \begin{cases} v_r^K(i+1,j+1) - v_r^K(i,j) & \text{if } k \in M, \\ v_r^K(i,j+1) - v_r^K(i,j) & \text{if } k \in N - M. \end{cases}$$

As in the proof of Theorem 3.1 we show that $\rho_r^K(i, j, k) \geqslant 0$ and $\rho_r^K(i, j, k) \leqslant \min\{\rho_r^K(i-1, j-1, k), \rho_r^K(i, j-1, k)\}$. For $0 \leqslant j < r - 1$, by (3.7) and (3.8), $\rho_r^K(i, j, k) = (K - r + 1)^{K-r}$. From (3.8), (3.9) and (3.10) $\rho_r^K(i, r-1, k) = v_1^{K-r+1}(1, 1) = (K - r + 1)^{K-r}$. We complete the proof by noting that for $j \geqslant r$, the desired inequalities follow from

$$\rho_r^K(i,j,k) = \rho_1^{K-r+1}(\max\{i-r+1, 0\}, j-r+1, k) \leqslant (K-r+1)^{K-r}$$

and the proof of Theorem 3.1.

Acknowledgement. Robert Bland, Gérard Cornuejols, William Pulleyblank and Michael Todd provided us with several helpful suggestions.

References

[1] Cornuéjols, G., Fisher, M. L. and Nemhauser, G. L. (1977). Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms. *Management Sci.* **23** 789–810.

[2] Karp, R. M. (1975). On the Computational Complexity of Combinatorial Problems. *Networks* **5** 45–68.

[3] Nemhauser, G. L., Wolsey, L. A. and Fisher, M. L. (July 1978). An Analysis of Algorithms for Maximizing Submodular Set Functions—I. *Math. Programming* **14** 265–294.

SCHOOL OF OPERATIONS RESEARCH AND INDUSTRIAL ENGINEERING, CORNELL UNIVERSITY, UPSON HALL, ITHACA, NEW YORK 14853