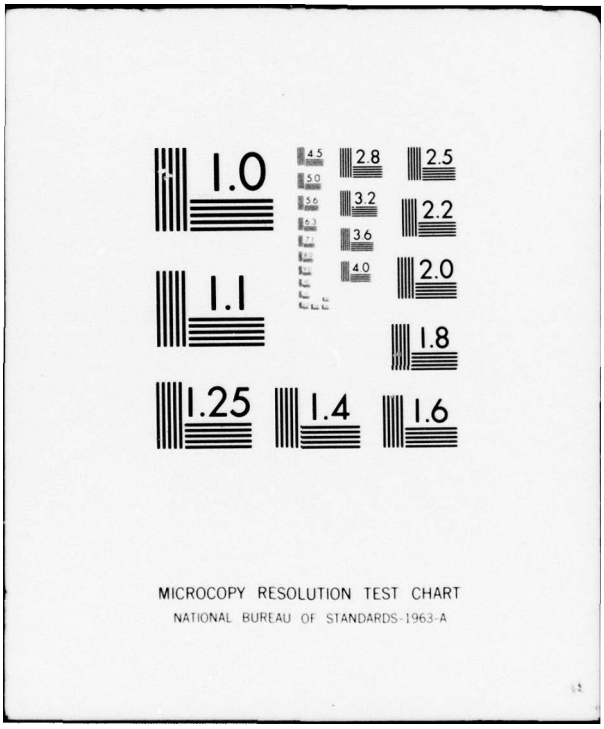
AD-A040 547 TEXAS UNIV AT AUSTIN CENTER FOR CYBERNETIC STUDIES F/G 5/1
AN APPLICATION OF QUASI-INTEGER PROGRAMMING TO A CAPITAL BUDGET--ETC(U)
MAR 77 R D ARMSTRONG, W D COOK N00014-75-C-0569
UNCLASSIFIED CCS-285 NL

| OF |
AD A040547

END

DATE FILMED
7-77


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

12 NW

CENTER FOR CYBERNETIC STUDIES

The University of Texas
Austin, Texas 78712

Research Report CCS 285

AN APPLICATION OF QUASI-INTEGER PROGRAMMING TO A CAPITAL BUDGETING PROBLEM IN PAVEMENT MAINTENANCE

by

R. D. Armstrong
W. D. Cook*
F. E. Palacios-Gomez

March 1977

*Faculty of Administrative Studies, York University, Toronto, Ontario.

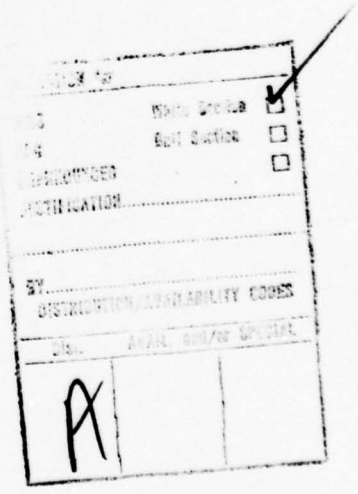
This research was partly supported by Project NR047-021, ONR Contract N00014-75-C-0569 with the Center for Cybernetic Studies, The University of Texas. Reproduction in whole or in part is permitted for any purpose of the United States Government.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
Business-Economics Building, 203E
The University of Texas
Austin, Texas 78712
(512) 471-1821

ABSTRACT

This paper models a capital budgeting problem in pavement maintenance as a nonlinear quasi-integer knapsack program and presents a solution procedure. It must be determined if certain segments of road will be repaved or major maintenance postponed for at least another year. If the road is to be repaved, a certain amount of variation in the funds expended is possible. The marginal return within the allowable interval of variation is estimated to be nonlinear. Also, a single linear constraint limiting the total amount of funds expended is present. Computational experience with the algorithm and a brief overview of other applications of the model are given.



1. Introduction

Capital budgeting problems in the operations research literature are frequently formulated as zero-one linear programming problems. However, it may occur that the "go or no go" restrictions characterized by binary constraints are too abrupt. Projects are often either not funded or funded within some interval. Modifications to a branch-and-bound algorithm to handle these "quasi-integer" constraints when returns are linear within the interval are given by Armstrong and Sinha [1]. In this paper we will consider a quasi-integer problem with nonlinear returns and a single linear budget constraint. This is a generalization of the classical binary knapsack problem.

The next section of this paper presents a detailed discussion of a quasi-integer nonlinear knapsack problem that the authors have encountered. Our particular application deals with the allocation of funds for pavement maintenance projects in the province of Ontario. A segment of road will either be completely repaved or major maintenance will be postponed for at least another year. If the road is to be repaved, a certain amount of variation in the funds expended is possible. Of course, when more funds are expended a higher quality road is constructed and a slower rate of deterioration takes place. The marginal value of expenditures beyond a base level is estimated to be nonlinear and a nonlinear quasi-integer knapsack problem arises. While we will be mainly interested in this application to pavement maintenance, it is apparent that other capital budgeting problems may be modeled in a similar manner.

Section 3 of this paper presents the theoretical foundation for the algorithm along with the general branch and bound concepts utilized. Section 4 discusses an implementation of the algorithm which parallels what we believe

to be the most efficient methods for solving the binary knapsack problem. Computational experience with a computer code version of the algorithm is given. Section 5 presents a sample pavement maintenance problem.

2. Applications

The quasi-integer knapsack model finds application in numerous areas involving capital budgeting on a single period basis. The pavement maintenance problem which gave rise to our research in this areas will be discussed in some detail but other applications are apparent.

In both USA and Canada, highway authorities are responsible for initiating effective maintenance strategies on pavements when required. Pavement maintenance is an area where millions of dollars are allotted annually for correction of defects. One particular highway management system with which the authors are familiar is the Ontario Ministry of Transportation and Communications. This organization maintains a vast bank of inventory (data) on the condition of all pavements in the province. The pavement condition rating (PCR) relates to the structural adequacy as well as to the geometrics. Structural adequacy is a function of the extent of cracking, roughness, frost heaving, settling, flushing and numerous other distortions. Geometrics have to do with the adequacy of lane width, slope and crossfall standards, shoulder width, ..., etc.

Studies conducted during the past fifteen years in the USA, Australia, Kenya, and Brazil (see for example [3], [12] and [13]) have provided information sources which are the principal inputs to decisions relating to highway rehabilitation. Data is available on pavement performance overtime, (relative to a variety of indices), under various rehabilitative alternatives. The alternatives available can, for purposes of illustration be classified as 1, 2 or 3 lifts of hot mix asphalt, reconstruction, recycling and hot mix patching.

By using data from the available studies as well as from historical records, pavement performance curves like the one illustrated in figure 1 have been obtained.

A family of functional forms which fit the data best and which were used in the analysis is the set of all exponentials of the form

$$P_o - P = KA^{\beta}$$

P_o is the rating immediately after, say, a 2 lift resurfacing, P is the rating A years after the resurfacing, and K and β are regression coefficients. This family of curves was used in analyzing pavement performance in the AASHO road tests [13].

In most instances once the highway has been placed into a given category according to traffic, number of lanes and region, the design (that is, the particular alternative of the ones listed above) is fixed. The aspect of rehabilitation which isn't predetermined, however, and about which we are concerned in this article, has to do with variation within a design. That is, once a design has been decided upon, up to 10 or 15% over the minimum cost necessary to meet standards can be spent on improving the future performance of a highway.

Different coefficients K and β can be generated from data on different levels of rehabilitation within the selected standard. Many different criteria exist for evaluating pavement performance. User cost, roughness, average PCR (per year), total PCR over the life of the pavement are examples. For purposes here we take the area under the PCR curve as a measure of worth of a pavement for a fixed analysis period of T years (e.g., T = 10) with some allowance for salvage to be made after that time.

For different levels of expenditure x_j on project j different performance curves arise; that is, for each x_j, appropriate K and β values can be determined. We assume x_j can lie in the range $[1, u_j]$ (x_j has been scaled by the minimum positive allocation). The worth of the pavement is then given by

$$f_j(x_j) = \int_0^T [\, P(x_j) + K(x_j)\, A^{\beta(x_j)}] \, dP + S_j(x_j) \tag{2.1}$$

where $S_j(x_j)$ is the salvage value.

There are many different ways of measuring the salvage value of a road. Generally, $S_j(x_j)$ should represent the long run worth of the road following the analysis period which ends at time T. Perhaps the most realistic approach in measuring $S_j(x_j)$ is to assume that an optimal maintenance procedure will be used after T. Then using standard discounting methods over a 20 or 30 year period following T, take $S_j(x_j)$ to be the best value which can be attained subject to budget restrictions. Salvage tables are currently available for roads in the various categories.

Let n denote the number of potential projects to be programmed. If $x_j = 0$ project j is not programmed; otherwise, $1 \le x_j \le u_j$. We therefore, wish to solve the quasi-integer programming problem

$$\text{Maximize} \quad \sum_{j=1}^{n} f_j(x_j),$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_j x_j \le B \tag{2.2}$$

$$1 \le x_j \le u_j \text{ or } x_j = 0, \; j = 1, 2, \ldots, n,$$

where B is the total budget and a_j is the minimum nonzero expenditure on project j.

The functions $f_j(x_j)$ take a form similar to that shown in figure 2. A quadratic provides an appropriate fit to the points generated from (2.1), so that the form of $f(x_j)$ assumed in (2.2) is

$$f(x_j) = b_j x_j^2 + c_j x_j + d_j \tag{2.3}$$

and $f(x_j) = 0$ for $x_j \notin [1, u_j]$.

In a later section we provide an example illustrating the use of these functions.

In the above we have described an application of the quasi-integer knapsack model for allocating funds towards pavement maintenance. Another example arises in defense construction. When funds are allocated to maintenance projects on a military base it is in many cases true that a variable amount of funds in some range can be allocated to a project. Different levels of improvements in the existing condition of electrical facilities can be made. Repairs to hangars and the associated facilities for aircraft maintenance can be initiated at different levels. So, the amount of money x_j allotted to project j is either 0, if no maintenance is carried out, or x_j lies in a closed interval. A measure of performance might be taken as the replacement value of the project.

In the following sections we present an algorithm for solving (2.2).

3. Algorithm for a Nonlinear Quasi-Integer Knapsack Problem

We will attempt to make the description of the algorithm as general as possible while taking full advantage of the structure associated with the pavement maintenance application. It is assumed that:

(i) $f_j(x_j) \geq 0$ in the closed interval $[1, u_j]$ and $f_j(x_j) = 0$ elsewhere. That is, for simplicity, the intercept term of (2.3) has been scaled appropriately so that the origin reflects the value of $f_j(0)$.

(ii) $f_j(x_j)$ is a monotone increasing concave function for $x_j \in [1, u_j]$.

(iii) $f_j(x_j)$ is differentiable for $x_j \in [1, u_j]$, where the derivative $f_j'(x_j)$ is defined to be the right derivative at 1 and the left derivative at u_j.

(iv) The inverse function of $f_j'(x_j)$, $1 \leq x_j \leq u_j$, is obtainable in closed form. That is, if $r_j(x_j)$ is the inverse function of $f_j'(x_j)$, then $y_j = f_j'(x_j)$ implies that $r_j(y_j) = x_j$.

Extensions of the algorithm to handle relaxations of assumptions (iii) and (iv) will be discussed in the conclusions. Notice that although we use the term "quasi-integer", no restriction is placed on the value of u_j other than it be greater than one. However, in our application, u_j will generally be between 1.05 and 1.15.

A branch-and-bound procedure [5,7] will be developed to solve problem (2.2). A convex relaxation of the problem will be solved at each node and the dichotomy performed will restrict some x_j to be 0 down one branch and to be in $[1, u_j]$ down the other branch. The branching process continues to split the solution space into smaller and smaller subsets until the optimal solution to the original problem is identified.

The relaxation is achieved by looking at each project on an individual basis and determining the largest return per dollar invested. This is given by

$$f_j(x_j^*)/a_j x_j^* = \text{maximum } f_j(x_j)/a_j x_j,\ j = 1, 2, \ldots, n.$$

To show that x_j^* can always be obtained through standard calculus procedures whenever our assumptions are satisfied, we prove the following theorem.

Theorem 1. The function $\theta(x) = f(x)/ax$ is quasi-concave in the interval $[1, u]$ when $f(x)$ is concave in this interval and $a > 0$.

Proof. Let

$$\Omega_\alpha = \{x \mid x \in [1, u],\ \theta(x) \geq \alpha\}$$

or

$$\Omega_\alpha = \{x \mid x \in [1, u],\ f(x) \geq \alpha ax\}$$

If $x \in \Omega_\alpha$ and $y \in \Omega_\alpha$, then $f(y) \geq \alpha ay$ and $f(x) \geq \alpha ax$. By the concavity of $f(x)$

$$f(x + (1 - \lambda)y) \geq \lambda f(x) + (1 - \lambda) f(y) \geq \alpha ax + (1 - \lambda)\alpha ay$$

$$= \alpha a(\lambda x + (1 - \lambda)y) = \alpha az, \text{ for } 0 \leq \lambda \leq 1.$$

Hence, $z = \lambda x + (1 - \lambda)y$ is in Ω_α and Ω_α is convex. By theorem 9.1.3 from Mangasarian [9] we have that $\theta(x)$ is quasi-concave.

Q.E.D.

Therefore, the point x_j^* is equal to either the left end point of the interval or can be obtained by taking the first derivative of $\theta_j(x_j)$, setting it equal to zero and solving. Assuming the latter situation holds, we have

$$f_j'(x_j^*)/a_j x_j^* - f_j(x_j^*)/a_j x_j^{*2} = 0$$

or

$$f_j'(x_j^*) = f_j(x_j^*)/x_j^*$$

Figure 3 presents two possible $f_j(x_j)$ and the corresponding x_j^*.

It will be assumed hereafter that the projects are ordered so that:

$$(3.1) \qquad f_1(x_1^*)/a_1x_1^* \geq f_2(x_2^*)/a_2x_2^* \geq \ldots \geq f_n(x_n^*)/a_nx_n^*$$

The same procedure is employed in the solution of the binary knapsack problem where projects are ordered in terms of best bang-for-buck.

We define $h_j(x_j)$ to be the line passing through the origin and the point $(x_j^*, f_j(x_j^*))$. Notice that $h_j(x_j)$ is a line tangent to $f_j(x_j)$ at x_j^* whenever x_j^* is a relative interior point of $[1, u_j]$.

By letting

$$g_j(x_j) = \begin{cases} h_j(x_j), & x_j < x_j^* \\ f_j(x_j), & x_j \geq x_j^* \end{cases}$$

we can write the following convexification of (2.2).

$$(3.2) \qquad \text{Maximize} \quad \sum_{j=1}^{n} g_j(x_j),$$

subject to

$$\sum_{j=1}^{n} a_jx_j \leq B$$

$$0 \leq x_j \leq u_j$$

Problem (3.2) is clearly a relaxation of (2.2). Hence, the optimal objective value for (3.1) provides an upper bound on the optimal objective value for (2.2). If an optimal solution to (3.2) is feasible for (2.2), then it is optimal for (2.2) as well. A similar relaxation is used by Miercort and Soland [10] for solving a discrete nonlinear problem via branch and bound.

To solve (3.2) we consider the Langrangian and perform a one dimensional search over the multiplier until the optimal solution (within a tolerance) is obtained. The following theorem which is analogous to a result for the binary knapsack problem gives a constructive procedure for determining an interval within which the optimal multiplier must be contained.

Theorem 2. An optimal solution to (3.2) exists which will have at most one component of the solution violating the constraints

$$x_j = 0 \text{ or } 1 \leq x_j \leq u_j, \; j = 1, 2, \ldots, n.$$

Proof. Let $\bar{\lambda}$ be the optimal multiplier for the Langrangean problem:

$$\underset{\lambda}{\text{Minimize}} \quad \underset{x}{\text{Maximize}} \quad \sum_{j=1}^{n} g_j(x_j) + \lambda \left(B - \sum_{j=1}^{n} a_j x_j\right),$$

subject to

$$0 \leq x_j \leq u_j, \; j = 1, 2, \ldots, n$$

$$\text{and } \lambda \geq 0.$$

From the Kuhn-Tucker conditions [9] the optimal solution $\bar{x}$ must satisfy

$$(3.3)\quad \bar{x}_j = \begin{cases} u_j \text{ if } g_j'(u_j)/a_j \geq \bar{\lambda} \\ r_j(a_j\bar{\lambda}) \text{ if } g_j'(u_j)/a_j < \bar{\lambda} < f(x_j^*)/a_j x_j^* \\ \text{a value in } [0, x_j^*] \text{ if } \bar{\lambda} = f(x_j^*)/a_j x_j^* \\ 0 \text{ if } \bar{\lambda} > f(x_j^*)/a_j x_j^* \end{cases}$$

$$\text{and} \quad \sum_{j=1}^{n} a_j \bar{x}_j = B \text{ if } \bar{\lambda} > 0,$$

where $r_j(\cdot)$ is the inverse function of the first derivative of $f_j(x_j)$. For the trivial case where $\bar{\lambda} = 0$, $\bar{x}_j = u_j$, $j = 1, 2, \ldots, n$ and the quasi-integer constraints are satisfied. Also, the quasi-integer constraint for variable j is always satisfied if $\bar{\lambda} \neq f(x_j^*)/a_j x_j^*$. Hence, the theorem will automatically hold unless $\bar{\lambda} = f(x_j^*)/a_j x_j^*$ for more than one j. In this case, we simply invest as much as possible in a project at a rate of return $\bar{\lambda}$ before investing in another project at this same rate.

Q.E.D.

Let $\bar{x}_r$ be the unique component of $\bar{x}$ (the optimal solution to the relaxation) which violates the quasi-integer constraint; that is, $0 < \bar{x}_r < 1$. Then $x_j^* \leq x_j \leq u_j$, $j < r$ and $x_j = 0$, $j > r$.

The branch-and-bound algorithm will create two new subproblems by dichotomizing as illustrated in figure 4. Therefore, at any stage of our algorithm we are solving a problem of the following form at node p.

$$\text{(3.4)} \qquad \text{Maximize} \quad \sum_{j \in JU} f_j(x_j) + \sum_{j \in RL} g_j(x_j)$$

subject to

$$\sum_{j \in JU} a_j x_j + \sum_{j \in RL} a_j x_j \leq B$$

and

$$1 \leq x_j \leq u_j, \; j \in JU$$
$$0 \leq x_j \leq u_j, \; j \in RL$$

where JU gives the index set of variables which are restricted by the branching process to be in $[1, u_j]$ and RL gives the index set of variables which have not been restricted as yet. There is a third index set JZ which gives the indices of the variables set to zero by the restrictions above.

The initial relaxation (3.2) has RL = {1, 2, ..., n}, JU = ϕ and JZ = ϕ, unless dominations within the solution set can be recognized. This subject will be discussed in the next section.

It is apparent that the result of theorem 2 can be extended to hold at any node. Again, an explicit representation of the optimal solution to (3.4) can be obtained in terms of the optimal multiplier. For completeness and because it will be useful later, we give Kunh-Tucker conditions which the optimal solution $\bar{x}^p$ at node p must satisfy in terms of the optimal multiplier $\bar{\lambda}^p$.

$$(3.5)\quad \bar{x}_j^p = \begin{cases} \text{The value determined by (3.3) for all } j \in RL \\ u_j;\ j \in JU,\ \bar{\lambda}^p \leq f_j'(u_j)/a_j \\ r_j(a_j\bar{\lambda}^p);\ j \in JU,\ f_j'(1)/a_j > \bar{\lambda}^p > f_j'(u_j)/a_j \\ 1;\ j \in JU,\ \bar{\lambda}^p \geq f_j'(1)/a_j \\ 0;\ j \in JZ \end{cases}$$

The fundamental steps of the algorithm have now been specified. The relaxation at each node is given by (3.4) and because of its convexity the above gives necessary and sufficient conditions to identify the optimal solution. Knowing that an optimal solution to (3.4) which is feasible for (2.2) provides a lower bound for (2.2) and that an optimal solution at any node provides an upper bound for all its descendants, a standard branch-and-bound algorithm such as that given by Geoffrion and Marsten [7] can be employed to solve (2.2). The next section describes how we have implemented the algorithm in a computer code.

4. Implementation and Computational Results

When implementing our algorithm we have tried to incorporate as many of the techniques used in solving binary knapsack problems as possible. As the result of improved pegging techniques and clever computer coding, very efficient codes have been written to solve the binary knapsack problem [2, 11, 15]. While many of the pegging procedures do not carry over to our problem, almost all of the list structures used to store the solution tree information are applicable with appropriate modifications. We begin by describing how we develop the solution tree and construct the relaxed candidate problems to be solved.

Both Barr and Ross [2], and Zoltners [15] report computational improvements in binary knapsack algorithms by utilizing a doubly-linked circular list. This greatly facilitates constructing and resolving relaxed candidate problems when a last-in-first-out (LIFO) branching rule is used. With regard to the nonlinear quasi-integer knapsack, rather than use an array with indicators to keep track of the index sets (for example, elements of the array may take of values 0, 1 or 2 to indicate that the corresponding variable is fixed at zero, fixed in the interval $[1, u_j]$ or free), a doubly-linked list and a singly-linked list [8] are used. The doubly-linked list is used to identify the free variables and is defined as follows for all $j \in RL$.

$$s(j) = \begin{cases} \text{minimum of } S_j, \text{ where } S_j \equiv \{k: k \in RL, k > j\} \\ \text{minimum of RL, if } S_j = \emptyset \end{cases}$$

$$p(j) = \begin{cases} \text{maximum of } P_j, \text{ where } P_j \equiv \{k: k \in RL, k < j\} \\ \text{maximum of RL, if } P_j = \emptyset. \end{cases}$$

A doubly-linked list is used because it is very important to maintain the ordering of variables as prescribed by (3.1). This ordering is not important with the variables associated with the JU index set. Thus, the index set JU may be represented as singly-linked list as follows for all $j \in JU$.

$$s(j) = \begin{cases} \text{index inserted in JU } \underline{\text{immediately prior}} \text{ to j's insertion in JU} \\ 0 \text{ if JU} = \emptyset \text{ at the time of j's insertion.} \end{cases}$$

When an index (say j) is to be removed from RL and placed in JU, the lists are updated as follows: $k = p(j)$, $i = s(j)$, $p(i) = k$, $s(k) = i$ $s(j) = m$ and $m = j$, where m is the last index to be inserted in JU and is initialized to zero. If j is later removed from JU, then $m = s(j)$. Because the predecessor of a variable which is removed from the RL list is not destroyed, it can be inserted back in the RL list quite easily. If j is to be inserted back in RL; $k = p(j)$, $i = s(k)$, $s(k) = j$, $s(j) = i$, and $p(i) = j$.

Through the use of these lists accessing of information in the desired order is expedited. A variable is considered when determining the values from (3.5) only when that variable is nonzero.

The algorithm, as coded, further restricts the unique variable x_j with a value in the interval $(0, x_j^*)$, first inspects the candidate problem with $1 \leq x_j \leq u_j$ and uses a LIFO procedure for choosing the candidate problem to consider when backtracking. The step-by-step method for solving (3.2) within the algorithmic framework will now be given.

STEP 1. Define $k = \{\text{minimum } \{i\} \mid \sum_{j=1}^{i} a_j u_j \geq B\}$, if such a k does not exist the optimal solution has $x_j = u_j$ for all j and the algorithm terminates; otherwise go to STEP 2.

STEP 2. Set $\bar{\lambda} = \lambda_k^*$

STEP 3. Determine $\bar{x}_j$, $j < k$, from equations (3.3) and define $S = \sum_{j=1}^{k-1} a_j \bar{x}_j$.

STEP 4. If $S \geq B$ (which can not occur the first time through), go to STEP 7; otherwise, go to STEP 5.

STEP 5. If $S + a_k x_k^* \geq B$, go to STEP 8; otherwise, $k = k + 1$ and go to STEP 6.

STEP 6. If $k > n$, set $\lambda_{k-1}^* = \lambda_k^*$ and $\lambda_k^* = 0$, go to STEP 7; otherwise, go to STEP 2.

STEP 7. The optimal λ for the relaxed candidate problem is between λ_{k-1}^* and λ_k^*, where λ_{-1} = maximum $f_j'(1)$. The optimal λ can then be determined by a sequential search over $\lambda_{k-1}^* \geq \lambda \geq \lambda_k^*$. The optimal solution will be feasible for the quasi-integer problem; hence, the algorithm terminates.

STEP 8. Create two branches in the solution tree by restricting $1 \leq x_k \leq u_k$ down one branch and $x_k = 0$ down the opposing branch.

The modifications to the above procedure are very minor when a relaxed candidate problem with the last restriction fixing $x_k = 0$ is to be solved. Because x_k is being removed from the problem and was not zero in the immediate successor, the optimal λ must be less than or equal to λ_k^*. Thus, the search can begin with $k = s(k)^*$ (where $s(k)^*$ is the successor of k before branching) and k is updated by placing $k = s(k)$ at each iteration. In STEP 3 equations (3.5) are used in place of (3.3) to determine $\bar{x}$. Also, the algorithm does not terminate when a feasible solution to the quasi-integer problem is found, but backtracks to determine either optimality or the next candidate problem to consider.

The modifications to the solution method when the last restriction places $1 \leq x_k \leq u_k$ are slightly more complicated and for clarity we will give

a step-by-step description. Because x_k is required to be nonzero in a feasible solution to candidate problem, the optimal λ for the immediate successor must be greater than or equal to λ_k^*.

STEP 1. Set $\bar{\lambda} = \lambda_k^*$.

STEP 2. Determine $\bar{x}_j$, $j \neq k$, from (3.5) and define

$$S = \sum_{j \in JU} a_j \bar{x}_j + \sum_{\substack{j \in RL \\ j \neq k}} a_j \bar{x}_j .$$

STEP 3. If $S \leq B$, go to STEP 5; otherwise go to STEP 4.

STEP 4. $k = p(k)$ Go to STEP 1.

STEP 5. If $S + a_k x_k^* \leq B$, go to STEP 7; otherwise, go to STEP 6.

STEP 6. Create two branches in the solution tree by restricting $1 \leq x_k \leq u_k$ down one branch and $x_k = 0$ down the opposing branch.

STEP 7. The optimal value for the relaxed candidate problem is between λ_k^* and λ_ℓ^*, where $\ell = s(k)$. The optimal $\bar{\lambda}$ can then be determined by a sequential search over $\lambda_k^* \geq \lambda \geq \lambda_\ell^*$. The optimal solution will be feasible for the quasi-integer problem; hence, the algorithm backtracks to determine either optimality or the next candidate problem to consider.

A very important property of our algorithm is that a continuous search over λ is required only when a feasible solution to the original problem is to be obtained. A bisecting search (see [14], P. 122) is used to find the optimal λ in this case. A deviation of 10^{-5} (a tolerance) from the complementary slackness condition is the criterion for terminating the search.

Certain bounds can easily be obtained for determining the suboptimality of a candidate problem and its' descendants. When a dichotomy is performed, a lower bound on the optimal solution is always obtainable by rounding to zero the x_k whose value is in the interval $[0, x_k^*]$. A lower bound is also available whenever a continuous search takes place. At any time, the algorithm maintains LB equal to the largest of these lower bounds (the incumbent objective function value). An upper bound on any feasible completion of the node currently under inspection is given by the optimal objective value of the relaxed candidate problem. Clearly, whenever this upper bound is not greater than LB, the node is fathomed and backtracking takes place.

Pegging (forced moves or flagging) is also possible in the quasi-integer nonlinear knapsack algorithm. Analogous to the binary knapsack problem where a variable is fixed at zero or one, in the process of solving the problem considered here a variable is fixed at zero or forced to be in $[1, u_j]$. To determine criteria as to when pegging can be performed, we state and prove at a nonrigorous level the following theorems.

Theorem 3. Define, $\bar{z}$, $\bar{x}$ and $\bar{\lambda}$ equal to the optimal objective value, x value, and λ value, of the relaxed candidate problem currently under consideration. If $j \in RL$ and $\lambda_j^* > \bar{\lambda}$, then no feasible descendent of this node will exist with $x_j = 0$ and an objective value greater than

$$UB_j = \bar{z} - (f_j(\bar{x}_j) - \bar{\lambda} a_j \bar{x}_j)$$

Proof.

If $x_j = 0$, which amounts to dropping x_j from the problem, it follows that the immediate loss in the objective value is $f_j(\bar{x}_j)$. On the other hand, an upper bound on the ultimate gain is obtained by investing all now available resource $a_j\bar{x}_j$ at the best possible rate of return $\bar{\lambda}$. Hence, an upper bound

on the objective value of any feasible descendant of this node with $x_j = 0$ is

$$\bar{z} - (f_j(\bar{x}_j) - \bar{\lambda} a_j \bar{x}_j)$$

Q.E.D.

Theorem 4. Let $\bar{z}$, $\bar{\lambda}$, and $\bar{x}$ be defined as in the previous theorem. If $j \in RL$ and $\lambda_j^* < \bar{\lambda}$, no feasible descendant of this node will exist with $x_j \geq 1$ and an objective value greater than

$$UB_j = \bar{z} - (\bar{\lambda} - \lambda_j^*) a_j$$

Proof.

Because $j \in RL$ and $\lambda_j^* < \bar{\lambda}$, $\bar{x}_j = 0$. If $\bar{x}_j \geq 1$ then at least a_j must be invested in project j with an upper bound on the rate of return given by λ_j^*. On the other hand, it will be necessary to cut back at least a_j from the projects currently having nonnegative investments. A lower bound on the rate of loss resulting from this cut back is $\bar{\lambda}$. It now follows that any feasible descendant of this node with $1 \leq x_j \leq u_j$ will have an objective value less than or equal to

$$UB_j = z - (\bar{\lambda} - \lambda_j^*) a_j$$

Q.E.D.

Whenever $UB \leq LB$, the index j can be removed from RL in all future descendants. The index j is placed in JU if UB_j is determined by the result of theorem 3 and is placed in JL if UB_j is determined by the result of theorem 4.

The algorithm described herein has been coded in FORTRAN and over a hundred randomly generated quasi-integer knapsack problems solved. The problems reported

in table 1 have u_j randomly generated from a uniform distribution between 1.01 and 1.11, the objective function parameters generated so that approximately 90% of all $x_j^* \in (1, u_j)$, and the a_j generated from uniform distributions with various upper and lower limits. All problems were solved on The University of Texas CDC 6600 Computer in a timesharing environment. Five problems were solved for each value of n. The times and number of branches indicate the average CPU seconds and branches, respectively, to varify optimality.

Comparing the results here and those reported by Barr and Ross [2], on a CDC CYBER 70/74-18, would indicate that a quasi-integer nonlinear knapsack requires approximately twenty times longer to solve than a binary knapsack with the same number of variables. As would be expected, the quasi-integer nonlinear knapsack algorithm generates fewer branches than the corresponding binary knapsack because the size of the feasible region has increased. On the other hand, the time required to solve the relaxed candidate problems is greater.

5. Sample Problem.

A sample of 400 projects was selected from records of the Ontario Ministry of Transportation. Associated with each project was a capital outlay x and, utilizing the subsequent performance related to each, f(x) was computed. This data was split into 20 categories according to the average daily traffic (ADT) (<2000 ADT, 2000-4000 ADT, 4000-8000 ADT and >8000 ADT) and region of the province (Northern, Northwestern, Eastern, Central and Southwestern). The number of data points varied from region to region. A regression was performed on the data in each of the categories, using a quadratic model which seemed to provide a good fit.

The results of the regression are shown in table 2. These values provide the objective function parameters for the quasi-integer knapsack problem. In the particular example solved as an illustration, the total budget available was taken as B = \$264, 590. The minimum expenditures a_j are also given in table 2. In the example we have taken each of the twenty functions as representing a specific project. This 20 project problem was then solved using the algorithm of sections 3 and 4. The results are shown in table 3.

In the general case with many projects there would be several in each category. All projects in the same category would be represented by the same quadratic, on a per mile basis. However, the scaling parameter, the minimum expenditure and the value f(x) (which is a function of length of the road section) would depend on the particular project in that category.

6. Conclusions

While we have directed our attention toward a capital budeting problem in pavement maintenance, it is apparent that other problems may be modeled in a similar fashion. Two assumptions concerning the general quasi-integer nonlinear knapsack problem were very important in the algorithmic development. The derivative of $f_j(x_j)$ was assumed to exist everywhere in $[1, u_j]$ and inverse function of this derivative was assumed to be obtainable in closed form. An absence of these two properties would greatly increase solution times as a search procedure would be required when solving each relaxation.

The computational results reported in this paper indicate that realistic sized capital budgeting problems can be solved as quasi-integer nonlinear knapsack problems. We have formulated a convexification of the original

problem and solved this relaxation at every stage of the algorithm. The strength of the method lies in the ability to solve the relaxations in an efficient manner. Hopefully, the techniques presented here can be utilized on other well-structured discontinuous nonlinear programming problems.

REFERENCES

1. Armstrong, R. D. and P. Sinha, "An Application of Quasi-Integer Programming to Menu Planning with Variable Portion Size," Management Science, Vol. 21, No. 4 (December 1974) 474-482.

2. Barr, R. S. and G. T. Ross, "A Linked List Data Structure for a Binary Knapsack Algorithm," Center for Cybernetic Studies, Report 232, August, 1975.

3. Claffey, P. J., "Running Costs of Motor Vehicles as Affected by Road Designs and Traffic," The National Academy of Sciences, 1971.

4. A. Charnes, and W.W. Cooper, Management Models and Industrial Applications of Linear Programming, Vol. I and II, John Wiley & Sons, Inc., New York, 1961.

5. Garfinkel, R. S. and G. L. Nemhauser, Integer Programming, John Wiley and Sons, New York, 1972.

6. Geoffrion, A., "Lagrangian Relaxation and Its Uses in Integer Programming," Mathematical Programming Study, 2, 1974, 82-114.

7. Geoffrion, A. and R. E. Marsten, "Integer Programming Algorithms: A Framework and State-of-the-Art Survey," Management Science, 18, 1972, 465-491.

8. Knuth, D. The Art of Computer Programming, Volume 1: Fundamental Algorithms, Addison - Wesley, Reading, Mass., (1970).

9. Mangasarian, O. L. Nonlinear Programming McGraw-Hill, Inc. New York 1969.

10. Miercort, F. A. and R. M. Soland, "Optimal Allocation of Missiles Against Area and Point Defenses," Operations Research, Vol. 19, No. 3, 1971, 605-617.

11. Nauss, R. M., "An Efficient Algorithm for the 0-1 Knapsack Problem," Management Science, Vol. 23, No. 1 (1976), pp. 27-31.

12. Pelensky, E., "The Cost of Urban Car Travel," Ontario Ministry of Transportation and Communications, 1970.

13. "The ASSHO Road Test," Highway Research Board of the National Academy of Sciences and the National Research Council, Report #5, Washington, D.C., 1962.

14. Zangwill, W. I., Nonlinear Programming: A Unified Approach, Prentice-Hall, Englewood Cliffs, N. J., 1969.

15. Zoltners, A., "A Direct Descent Binary Knapsack Algorithm," Working Paper No. 75-31, School of Business Administration, University of Massachusetts, Amherst 1975.

n	50	100	150	200	250	300	350	400	450	500
Time (seconds)	.20	.46	.62	.56	.86	1.29	1.20	1.22	3.00	2.34
Branches	63	60	59	34	42	58	35	41	99	55

Table 1. Five randomly generated quasi-integer nonlinear knapsack problems were solved for each value of n. The average time and average number of branches for each of the problem sets are indicated.

raffic	North				Northwestern				Eastern				Central				Southwestern			
	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d	a	b	c	d
2000	25.4	-3.0	10.0	-3.04	32.3	-3.1	28.0	-3.35	23.0	-2.8	7.6	3.33	28.0	-2.1	7.5	-2.17	41.3	-3.9	8.4	-4.55
000- 000	27.1	-5.0	12.0	-5.91	31.9	-4.2	10.3	-4.79	46.1	-5.3	17.0	-6.27	40.3	-3.8	12.0	-4.58	36.7	-3.6	11.0	-3.9
000 000	25.5	-3.6	8.3	-3.92	24.8	-3.0	7.0	-3.49	36.7	-4.2	12.0	-4.46	33.3	-5.3	12.4	-6.12	19.5	-1.3	4.0	-1.37
8000	31.8	-2.7	9.4	-2.84	28.8	-4.1	9.0	-4.2	32.2	-3.8	9.0	-4.14	26.1	-2.5	8.7	-2.9	38.4	-2.7	9.3	-2.7

Table 2. Data is given from a regional planning model in pavement maintenance for the Province of Ontario. Objective functions are of the form $bx^2 + cx + d$ and the single linear constraint has coefficient a (in thousands of dollars). The upper bound on the variables is 1.1.

Traffic	North	Northwestern	Eastern	Central	Southwestern
<2000	1.1	1.1	0.	1.04	0.
2000 -4000	0.	0.	1.1	0.	0.
4000 -8000	0.	0.	0.	0.	1.0
>8000	1.09	0.	0.	1.1	1.0

Table 3. This table gives the optimal solution to the model described in Table 2.

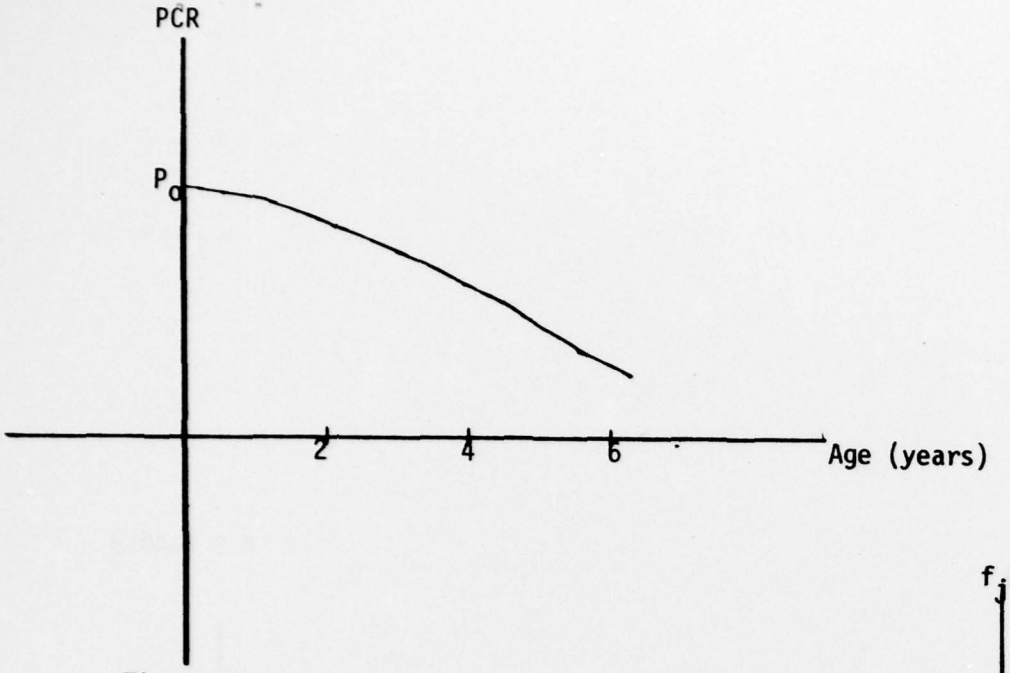


Figure 1

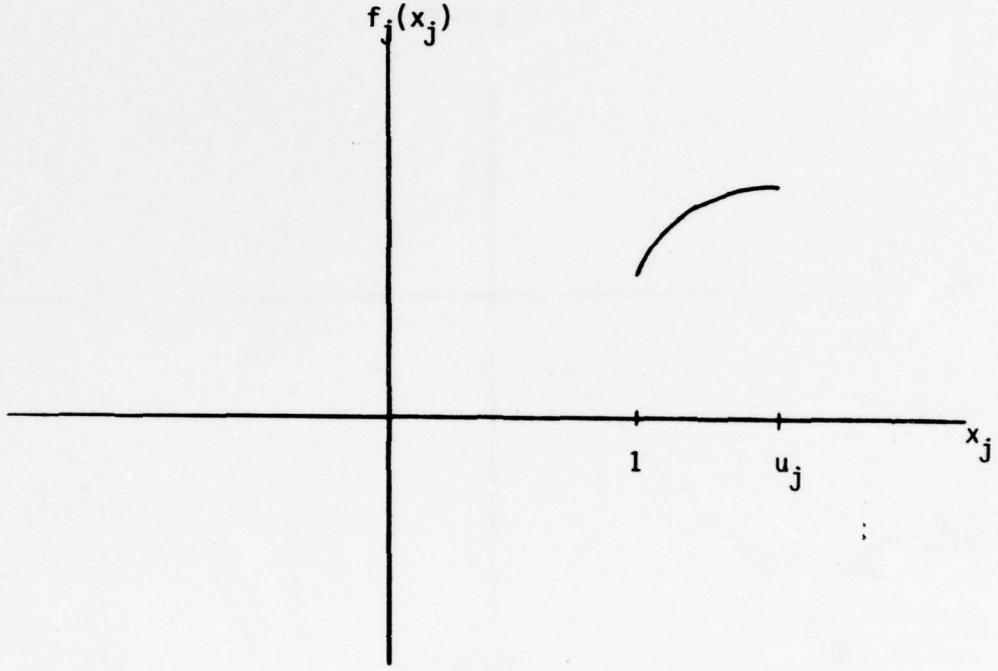


Figure 2

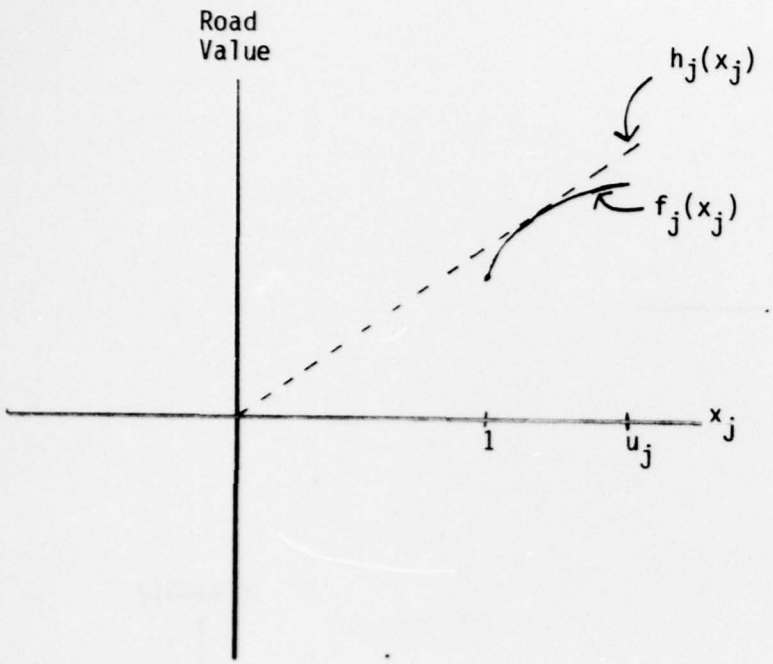


Figure 3 a

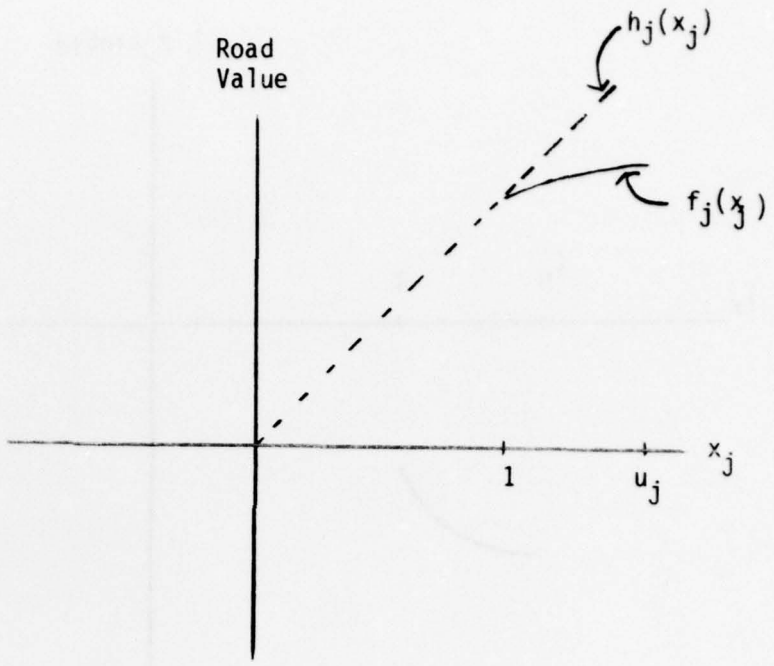


Figure 3 b

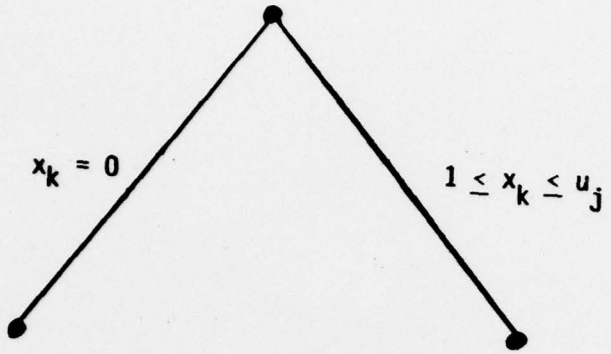


Figure 4

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
Center for Cybernetic Studies
The University of Texas

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
An Application of Quasi-Integer Programming to a Capital Budgeting Problem in Pavement Maintenance

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)
Research Rept.

5. AUTHOR(S) (First name, middle initial, last name)
Ronald D. Armstrong
W. D. Cook
Fernando E. Palacios-Gomez

6. REPORT DATE
March 1977

7a. TOTAL NO. OF PAGES
27

7b. NO. OF REFS
15

8a. CONTRACT OR GRANT NO.

b. PROJECT NO.

c.

d.

32p.

9a. ORIGINATOR'S REPORT NUMBER(S)
Center for Cybernetic Studies
Research Report CCS-285

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT
This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES
N00014-75-C-0569

12. SPONSORING MILITARY ACTIVITY
Office of Naval Research (Code 434)
Washington, D.C.

13. ABSTRACT

This paper models a capital budgeting problem in pavement maintenance as a nonlinear quasi-integer knapsack program and presents a solution procedure. It must be determined if certain segments of road will be repaved or major maintenance postponed for at least another year. If the road is to be repaved, a certain amount of variation in the funds expended is possible. The marginal return within the allowable interval of variation is estimated to be nonlinear. Also, a single linear constraint limiting the total amount of funds expended is present. Computational experience with the algorithm and a brief overview of other applications of the model are given.

DD FORM 1 NOV 68 1473 (PAGE 1)
S/N 0101-807-6811

406 197

Unclassified
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Knapsack problems						
Quasi-Integer Programming						
Capital Budgeting						
Pavement Maintenance						