Mathematical Programming 12 (1980) 132–149.
North-Holland Publishing Company

QUADRATIC KNAPSACK PROBLEMS*

G. GALLO

Istituto M. Picone per le Applicazioni del Calcolo, CNR, Roma, Italy

P.L. HAMMER and B. SIMEONE

University of Waterloo, Waterloo, Ont., Canada

Received 17 June 1977
Revised manuscript received 10 April 1978

The quadratic knapsack (QK) model naturally arises in a variety of problems in operations research, statistics and combinatorics. Some "upper planes" for the QK problem are derived, and their different uses in a branch-and-bound scheme for solving such a problem are discussed. Some theoretical results concerning the class of all upper planes, as well as extensive computational experience, are reported.

Key words: Knapsack Problem, Quadratic Programming, Upper Planes, Branch-and-Bound, Computation.

1. Introduction

1.1. Let B^n be the set of all 0–1 n-vectors, and define as a *quadratic* 0–1 *knapsack problem* (QK) any of the following two problems:

$$\begin{aligned} \max \quad & x^{\mathrm{T}}Qx, \\ \text{s.t.} \quad & ax \leq b, \\ & x \in B^n, \end{aligned} \tag{1.1}$$

$$\begin{aligned} \min \quad & x^{\mathrm{T}}Qx, \\ \text{s.t.} \quad & ax \geq b, \\ & x \in B^n, \end{aligned} \tag{1.2}$$

where Q is a non-negative[1] square matrix of order n, a is a positive n-vector and b a positive scalar. Since $x_i = x_i^2$ for $x_i = 0, 1$ linear terms in the objective function of (1.1) or (1.2) can be implicitly taken into account in the quadratic part. We can also assume that Q is symmetric. This model arises from a variety of applications.

* Presented at the IX International Symposium on Mathematical Programming, Budapest, August 1976.

[1] Actually, the presence of negative elements on the diagonal does not affect our development.

Example 1. (Witzgall [14]). The technology of communication satellites has recently inspired the design of mail subsystems in which messages are transmitted electronically rather than physically. Electronic message handling stations convert physical messages into electronic ones and vice versa, and communicate with each other through satellites. Known n potential sites for the stations, the investment cost a_i for building a station in site i, and the average daily mail volume q_{ij} observed between i and j, the problem of selecting a set $S \subseteq \{1, \ldots, N\}$ of locations such that the global traffic $\sum_{i,j \in S} q_{ij}$ is maximized and a budget constraint $\sum_{i \in S} a_i \leq b$ is met is obviously of the form (1.1).

Similar problems have been investigated regarding the location of railway stations (Land [8]), freight handling terminals (Rhys [11]) and airports.

Example 2. In hydrological studies, the observations made by different pluviometers in a same region are usually seen to be correlated. Thus it is desirable to reduce the number of puviometers by choosing only a few of them, with minimal redundancy, and an acceptable loss of information. If q_{ij}, $i \neq j$, is the absolute value of the covariance between the rainfall in stations i and j, estimated from a record of past observations, $q_{ii} = 0$, a_i is the variance of rainfall in i, and b is a fixed fraction of the total variance $\sum_i a_i$, the problem can be formulated as (1.2). A similar model arises in applications to portfolio selections (Laughunn [9]), when one wishes to select a subset of n possible investment proposals such that some fixed minimum return is guaranteed and the risk is minimized. The sum of the absolute values of the covariances of the (random) returns associated to the proposals can be taken as a reasonable measure of risk.

Example 3. The problem of determining whether a given graph possesses a clique of order k (k is a fixed integer) is a special case of (1.1), where Q is the node–node adjacency matrix of the graph, $a_i = 1$ for all i and $b = k$. The clique does exist if and only if the optimum value of (1.1) is $k(k-1)$. It is well-known (Karp [7]) that this clique problem is NP-complete. More generally, the QK problem can be formulated in the following graph-theoretic terms. Given a graph G, let a "benefit" be associated with each edge, and a "cost" with each node of G. Define the "benefit" of a subgraph H to be the sum of the benefits of all edges in H, and the "cost" of H to be the sum of the costs of all nodes in H. (1.1) is then equivalent to the problem of finding a subset S of nodes, such that the subgraph induced by S has a cost not exceeding some fixed level, and yields a maximum benefit.

1.2. The present paper is mainly concerned with the formulation (1.1) of QK. Actually, the algorithms we propose can be converted, in a straightforward way, for handling problem (1.2).

Any lagrangian relaxation of (1.1) with respect to the knapsack constraint is amenable to the maximization over B^n of a quadratic function whose diagonal

terms are negative and whose off-diagonal terms are non-negative. Rys [11] showed that the last problem is equivalent to a network flow one and Balinski [1] proposed a labelling procedure for solving it.

1.3. The algorithms developed in this paper are based on bounds on the optimum value of (1.1), obtained by considering linear relaxations in which the quadratic objective $f(x)$ is replaced by an "upper plane",[2] i.e., a linear function $g(x)$ which dominates $f(x)$ in all feasible points. This approach is motivated by the easiness with which even large linear knapsack problems can be solved (Salkin and de Kluyver [19]). The use of upper planes is not new in the literature (Taha [13]) and goes back to at least as early as to the works of Beale [3] and Balinski [1] on certain nonlinear transportation problems. Upper planes have usually been considered in connection with the outer-linearization of a concave function; also, Cabot and Francis [4] suggested a simple method for deriving upper planes for non-convex continuous quadratic problems. (As a matter of fact, the upper planes constructed in Section 2 of this paper turn out to be closely related to those of Cabot and Francis.) The focus of this work is on the relative efficiency of different upper planes and on the different ways in which they can be exploited in branch-and-bound techniques for solving (1.1). Extensive computational experience is presented. Finally, in the last section some theoretical properties concerning the family of *all* upper planes are investigated.

2. Upper planes for the quadratic knapsack problem

2.1. Let us denote by $X \equiv \{x \in B^n: ax \leq b\}$ the feasible set of (1.1). Without loss of generality, we can assume $\sum_{i=1}^{n} a_i \; a_1 > b > a_1 \geq a_2 \geq \cdots \geq a_n > 0$. An *upper plane* (UP) for the function $f(x) = x^T Q x$ in X is any linear function $g(x)$ such that $g(x) \geq f(x)$ for all $x \in X$. Given an UP $g(x)$ for $f(x)$ in X, the corresponding *linear relaxation* of QK is the (linear) knapsack problem

$$\max\{g(x): x \in X\}. \tag{2.1}$$

From the solution of (2.1) both an upper and a lower bound on the optimum value of (1.1) can be derived. Indeed, if x^* and $\hat{x}$ are optimal solutions of (1.1) and (2.1) respectively, one has $f(\hat{x}) \leq f(x^*) \leq g(\hat{x})$.

2.2. Given the upper plane $g(x)$, a cheaper way of getting bounds consists in solving the *continuous* knapsack problem

$$\max\{g(x): x \in \hat{X}\}, \tag{2.2}$$

where $\hat{X} \equiv \{x \in R^n: ax \leq B, O \leq x \leq e\}$, with $e = (1, 1, \ldots, 1)$.

[2] Some authors use the term "linear overestimator".

Such a problem admits an optimal solution whose components are all zeroes or one, with the possible exception of a single component. Setting this variable down to zero, a binary feasible vector x^0 is obtained and $f(x^0)$ is a lower bound for the quadratic optimum; an upper bound is given by the optimum value of (2.2).

2.3. The algorithm which will be described in the next section is an enumerative one, in which the exploration of each node can be viewed as a two-stage process. The rôle of the first stage is the generation (through the exact or approximate solution of a number of linear knapsack problems) of the coefficients of an upper plane, while the rôle of the second stage is the determination of upper and lower bounds on the maximum of $f(x)$ in X (through the solution of the linear relaxation corresponding to the above upper plane).

2.4. A simple way to derive upper planes is the following. If v_j is an upper bound of the set $\{q_j^T x: x \in X\}$, where q_j is the jth column of Q, the function $\sum_{j=1}^n v_j x_j$ is obviously an UP for $f(x) = \sum_{j=1}^n (q_j^T x) x_j$ in X. This turns out to be true even when v_j is an upper bound for the set $\{q_j^T x: x \in X, x_j = 1\}$, since $(q_j^T x)x_j \leq v_j x_j$ when $x_j = 1$ and $(q_j^T x)x_j = v_j x_j = 0$ when $x_j = 0$.

Four possible choices for v_j are given, under the hypothesis that $q_{ij} \geq 0$ for all i, j:

$$v_j^1 = \sum_i q_{ij},$$

$$v_j^2 = \sum_i^{(h)} q_{ij}, \tag{2}$$

where $\sum_i^{(h)}$ means that the sum is restricted to the h largest elements of the jth column of Q, and h is the maximum number of ones in a feasible solution (since the a_j's are non-increasing, h is the largest index k for which $a_{n-k+1} + \cdots + a_n \leq b$),

$$v_j^3 = \max\{q_j^T x: x \in \hat{X}\}, \tag{3}$$

(of course one can take the integer part of the r.h.s. if Q is integer),

$$v_j^4 = \max\{q_j^T x: x \in X\}. \tag{4}$$

The order relationship between the four upper bounds is shown in Fig. 1, where a continuous arrow points from a smaller to a larger upper bound, while the dotted arrow means that in most observed cases (but not always) v_j^3 was found to be smaller than v_j^2.[3]

On the other hand, the computational effort needed for evaluating the v_j's increases from (1) to (4). The upper bound (4) is the best possible one, but it is also by far the most expensive. Extensive numerical experimentation has been

[3] A remarkable exception arises when all the maximal feasible vectors have the same number of ones. In this case $v_j^2 = v_j^4 \geq v_j^3$.

Fig. 1.

performed (see Section 4) in order to find an upper plane which corresponds to a good trade-off between computational effort and tightness to the quadratic function.

2.5. A given lower bound on the optimum value of (1.1) can often be improved, at low cost, through a sequence of the following elementary operations (suggested for linear knapsack problems by Petersen [10]):

Fill-up, which transforms a given feasible point x, having $x_j = 0$, to a point of the form $x' = x + u_j$ (u_j is the jth unit vector). Then one has $f(x') \geq f(x)$, the inequality being strict unless q_j is the zero vector.

Exchange, which transforms a point x such that $x_i = 1$ and $x_j = 0$ for some i, j to the point $x' = x - u_i + u_j$, so that $x'_i = 0$ and $x'_j = 1$. The exchange is performed only if x' is still feasible and (using the terminology in [5]) the *second derivative*

$$\Delta_{ij}(x) = f\Big(x_1, \dots, \overset{i}{\underset{\downarrow}{1}}, \dots, \overset{j}{\underset{\downarrow}{0}}, \dots, x_n\Big) - f\Big(x_1, \dots, \overset{i}{\underset{\downarrow}{0}}, \dots, \overset{j}{\underset{\downarrow}{1}}, \dots, x_n\Big)$$

is negative. Since

$$\Delta_{ij}(x) = \Delta_i(x) - \Delta_j(x) + 2q_{ij}(x_i - x_j),$$

where

$$\Delta_k(x) = f\Big(x_1, \dots, \overset{k}{\underset{\downarrow}{1}}, \dots, x_n\Big) - f\Big(x_1, \dots, \overset{k}{\underset{\downarrow}{0}}, \dots, x_n\Big) = q_{kk} + 2\sum_{h \neq k} q_{hk}x_h$$

is the *first derivative*, only the first derivatives need to be kept in storage; they can be updated after each elementary operation by simple formulae.

There are two good reasons for trying to improve the lower bounds. As we shall see later, the above discussed linear relaxations yield very good *lower* bounds. Thus, the solution of the linear relaxation, followed by the improving procedure sketched above, gives rise to an economic and effective *heuristic algorithm*, which often leads to the true optimum. If, on the other hand, one wishes to use linear relaxations in a branch-and-bound scheme for finding an optimal solution, early knowledge of tight lower bounds enhances the fathoming power of the algorithm.

3. Use of upper planes in branch-and-bound algorithms

3.1. Upper planes can be exploited as basic tools in deriving enumerative algorithms for QK. For any subproblem obtained from (1.1) by restricting certain variables to 0 or 1, it is possible to derive bounds through the solution of a discrete or continuous linear relaxation. Moreover, upper planes play an interesting rôle in other basic aspects of enumerative algorithms, such as forcing of variables and branching. Up and down penalties for linear relaxations can also be obtained with a modest cost. Finally, the amount of work required in order to generate the linear relaxations can be considerably reduced by means of recursive formulae.

Throughout this section, we shall assume, for the sake of simplicity, that all the data Q, a, b are integer; we shall denote by U, Z and F the index sets of the variables which, in the current subproblem (P), are fixed to 1, 0 or are free, respectively, so that the feasible set in (P) is

$$S \equiv \{x: x \in X;\ x_j = 1\ \forall j \in U:\ x_j = 0\ \forall j \in Z\};$$

$v_0 + \sum_{j \in F} v_j x_j$ will stand for the current UP, and $\sum_{j \in F} a_j x_j \leq \bar{b}$ for the current knapsack constraint, while (P_r^0) and (P_r^1) will represent two subproblems, with feasible sets S_r^0 and S_r^1 respectively, obtained from (P) by adding the further constraint $x_r = 0$ or $x_r = 1$.

3.2. *Bounding.* In each subproblem (P), the restriction of the objective function f to S is a non-negative quadratic function in the free variables, for which we can construct an upper plane $v_0 + \sum_{j \in F} v_j x_j$, according to one of the rules (1), ..., (4) of Section 2.4. It is important to recognize that, in all four cases, the law which associates to any subproblem (P) the corresponding upper plane g is *adaptive*: in other words, if g' and g'' are the upper planes corresponding to (P_r^0) and (P_r^1), generated according to the *same* rule as for g, one has $g'(x) \leq g(x)\ \forall x \in S_r^0$, and $g''(x) \leq g(x)\ \forall x \in S_r^1$. Let us prove it, for illustration purposes, in the case that $g(x)$ is the UP $v^4 x$. Let us also restrict our attention to the upper plane $g'(x) = v'^4 x$ for (P_r^1). Dropping for simplicity the index "4", we have:

$$v_0 = \sum_{i,j \in U} q_{ij}, \qquad v_j = 2 \sum_{i \in U} q_{ij} + \max\left\{ \sum_{i \in F} q_{ij} x_i : \sum_{i \in F} a_i x_i \leq \bar{b},\ x_i = 0, 1 \right\},$$

$$v_0' = \sum_{i,j \in U} q_{ij} + q_{rr},$$

$$v_j' = 2 \sum_{i \in U} q_{ij} + 2q_{rj} + \max\left\{ \sum_{i \in F - \{r\}} q_{ij} x_i : \sum_{i \in F - \{r\}} a_i x_i \leq \bar{b} - a_r,\ x_i = 0, 1 \right\}.$$

Hence we have

$$v_0 = v_0' - q_{rr}, \qquad v_j \geq q_{rj} + (v_j' - 2q_{rj}) = v_j' - q_{rj}, \quad \forall j \in F - \{r\},$$

so that, for any $x \in S_r^1$,

$$g(x) = v_0 + v_r + \sum_{j \in F-\{r\}} v_j x_j \geq (v_0' - q_{rr}) + \left(q_{rr} + \sum_{i \in F-\{r\}} q_{ir} x_i\right)$$
$$+ \left(\sum_{j \in F-\{r\}} v_j' x_j - \sum_{j \in F-\{r\}} q_{rj} x_j\right) = v_0' + \sum_{j \in F-\{r\}} v_j' x_j = g'(x),$$

3.3. *Branching.* Let x^0 be an optimal solution for the linear relaxation of the current subproblem, and let $F_1 \equiv \{j \in F: x_j^0 = 1\}$. A reasonable criterion for selecting the branching variable x_r is to choose r such that

$$\frac{v_r}{a_r} = \max_{j \in F_1} \frac{v_j}{a_j},$$

the branch $x_r = 0$ being explored first.

3.4. *Recursions.* If $v_0^1 + \sum_{j \in F} v_j^1 x_j$ is the UP for the current node, generated according to rule (1) of Section 2.4, the corresponding UPs for the two successors can be generated through simple recursions. Actually, a direct computation shows that:

Proposition 3.1. *The UPs of type* (1) *for* (P_r^0) *and* (P_r^1) *are given by*

$$v_0^1 + \sum_{j \in F-\{r\}} (v_j^1 - q_{rj}) x_j$$

and by

$$\left(v_0^1 + q_{rr} + \sum_{j \in U} q_{rj}\right) + \sum_{j \in F-\{r\}} (v_j^1 + q_{rj}) x_j$$

respectively.

Slightly more complicated recursions hold for the UPs (2) and (3). Even if recursions are not used, it is important to notice that the most expensive part of the work required for generating the UP (3), i.e., sorting the elements of each column j according to non-increasing ratios q_{ij}/a_i, need not be repeated, once it has been done for the initial one. A similar observation can be done for the UP (2).

3.5. *Forcing.* At each node, one is really interested only in those feasible points x in which the quadratic objective function, and hence, a fortiori, the current UP, is strictly greater than the current lower bound $\underline{z}$. The constraint $v_0 + \sum_{j \in F} v_j x_j \geq \underline{z} + 1$, combined with the knapsack constraint $\sum_{j \in F} a_j x_j \leq \bar{b}$, may be very effective in forcing variables to the value 0 or 1. To this end, the constraint pairing techniques described in Hammer et al. [6] can be successfully exploited. Such techniques also allow for detecting whether the system composed by the two above inequalities is inconsistent.

3.6. *Penalties.* Let u, u_r^0 and u_r^1 denote upper bounds for the subproblems (P), (P_r^0), (P_r^1) respectively, computed according to one of the methods of Section 2.4.

Down and *up penalties*, i.e., underestimates of the decreases $u - u_r^0$ and $u - u_r^1$ can be exploited, in the usual way, for fathoming (P) by bound, for forcing variables or – as an alternative to the criterion in Section 3.3 – for selecting the branching variable.

The following proposition provides an inexpensive procedure for evaluating down and up penalties when the upper bounds are obtained by solving *continuous* linear relaxations. Such penalties are valid for an arbitrary upper plane, provided that it is *adaptive* (according to the definition in Section 3.2).

Let $v_0 + \sum_{j \in F} v_j x_j$ be such an UP for (P). Assuming the free variables numbered according to non-increasing ratios v_j/a_j, let x_l be the last positive variable in the optimal solution $\hat{x}$ of the corresponding continuous linear relaxation of (P).

Proposition 3.2. (a) *If* $\hat{x}_r = 1$,

$$\left(v_r - \frac{v_l}{a_l} a_r\right)$$

is a down penalty for x_r.

(b) *If* $\hat{x}_r = 0$,

$$\left(\frac{v_l}{a_l} a_r - v_r\right)$$

is an up penalty for x_r.

(c) *If* $0 < \hat{x}_r < 1$ (i.e., $l \equiv r$),

$$\hat{x}_l\left(v_l - \frac{v_{l+1}}{a_{l+1}} a_l\right) \quad \text{and} \quad (1 - \hat{x}_l)\left(\frac{v_{l-1}}{a_{l-1}} a_l - v_l\right)$$

are down and up penalties for x_r, *respectively.*

Let us prove, for example, (a). If $\hat{x}_r = 1$, since the upper plane is adaptive we have

$$u_r^0 \leq \max\left\{v_0 + \sum_{j \in F} v_j x_j : \sum_{j \in F} a_j x_j \leq \bar{b};\ 0 \leq x_j \leq 1, \forall j \in F;\ x_r = 0\right\}. \tag{3.1}$$

On the other hand,

$$u = v_r + \max\left\{v_0 + \sum_{j \in F-\{r\}} v_j x_j : \sum_{j \in F-\{r\}} a_j x_j \leq \bar{b} - a_r;\ 0 \leq x_j \leq 1, \forall j \in F - \{r\}\right\}$$

$$\geq v_r - \frac{v_l}{a_l} a_r + \max\left\{v_0 + \sum_{j \in F-\{r\}} v_j x_j : \sum_{j \in F-\{r\}} a_j x_j \leq \bar{b};\ 0 \leq x_j \leq 1, \forall j \in F - \{r\}\right\},$$

that is,

$$u \geq v_r - \frac{v_l}{a_l} a_r + \max\left\{ v_0 + \sum_{j \in F} v_j x_j : \sum_{j \in F} a_j x_j \leq \bar{b}; 0 \leq x_j \leq 1, \forall j \in F; x_r = 0 \right\}.$$

Comparing the last inequality with (3.1), one gets

$$u - u_r^0 \geq v_r - \frac{v_l}{a_l} a_r,$$

which proves (a). Statements (b) and (c) can be proved in a similar way.

3.7. We now give a general description of a class of branch-and-bound methods for QK, involving the procedures discussed in this section. Two algorithms in the class may differ in one or more of the following aspects:

— the type of upper plane ((1), (2), (3) or (4) of Section 2.2) generated at every node during the first stage (see Section 2.1).

— the kind (discrete or continuous) of linear relaxation solved at every node during the second stage.

— the presence or absence of routines for constraint pairing and for evaluating and using penalties.

Each algorithm performs a standard search over a binary tree, whose nodes correspond to subproblems obtained by restricting some variables to the values 0 or 1. The two successors of each node are generated by fixing some free variable to 0 or to 1, respectively. In exploring the tree, a LIFO priority rule is adopted. At each node the following steps are performed:

Step 1: A linear relaxation for the corresponding subproblem is generated.

Step 2: An attempt is made in order either to detect infeasibility or to force a single variable by the constraint pairing techniques mentioned in Section 3.4. If the problem is feasible and no variable can be forced,

Step 3: The relaxed subproblem is solved and the bounds on the quadratic optimum are updated.

Step 4: Every time the lower bound is improved, use is made of the routine described in Section 2.5 in an attempt to further improving it.

Step 5: If the subproblem cannot be fathomed by bound or by optimality,

Step 6: Up and down penalties for all the free variables are computed, by means of which the algorithm again tries either to fathom the subproblem by bound or to force as many variables as possible. In case of failure,

Step 7: A branching is made on the variable with the highest up or down penalty.

Step 8: (Only if Steps 6 and 7 are not executed) A branching is made on the variable selected according to the criterion of Section 3.

Step 9: Whenever a subproblem is fathomed (by infeasibility, bound or

optimality), backtracking to the predecessor occurs and the other branch emanating from it is explored, unless this has already been done.

Steps 2, 6, 7 are optional.

4. Computational experience

4.1. Numerical experimentation has been performed on ten different variants of the general branch-and-bound method sketched at the end of the last section. To this end, a FORTRAN code for this class of algorithms has been implemented on an IBM 370/168.

For reference, each variant is identified by a label, in which the first character is B or C, according to whether discrete or continuous linear relaxations are solved during the second stage at each node; the second character is 1, 2, 3 or 4, according to the type of upper plane generated during the first stage; one or more optional characters may follow, with the following meaning:

F a routine for forcing variables, based on constraint pairing techniques, is included.

F_k as above, but only k surrogate constraints are generated rather than $n+2$ as suggested in [6].

P a routine for evaluating and using penalties is included.

4.2. The numerical experiments were performed on randomly generated test problems, ranging from 10 to 70 variables, in which the elements of Q were integers uniformly distributed between 0 and 100, the knapsack coefficients a_j were integers uniformly distributed between 1 and 50, and the b's were integers uniformly distributed between 1 and $\Sigma_{j=1}^{n} a_j$. Unless otherwise specified, Q was taken full (100% dense). For each problem size $n = 10, 20, \ldots$, ten test problems were run.

4.3. Let UB and LB denote the upper and lower bounds for the initial subproblem, computed as in 2.1 and 2.2, and let OPT denote the quadratic optimum. Table 1 gives, for the variants B1, B2, B3, B4, C3, the average "Av", the standard deviation "Sd" and the maximum "Max" (estimated, for each size n, from a sample of ten test problems) of $(UB - LB)/LB\%$ and of $(OPT - LB)/LB\%$.

The following remarks are in order:

— The upper bounds given by $B4$ and $B3$ are remarkably better than those provided by $B2$ and $B1$.

— The lower bounds are always much closer to the quadratic optimum than the upper bounds.

— The relative error of the bounds with respect to the optimum appears to remain of the same order of magnitude as the size n of the problem increases.

Table 1

Version	n	$\frac{UB - LB}{LB}\%$ Av	Sd	Max	$\frac{OPT - LB}{LB}\%$ Av	Sd	Max
$B1$	10	75.42	82.77	293.48	8.52	16.16	51.55
	20	166.02	229.03	835.05	1.03	2.45	8.30
	30	74.01	92.09	307.47	0.63	1.69	5.68
$B2$	10	25.73	14.30	42.24	3.80	5.73	15.09
	20	39.61	22.97	76.46	3.82	8.28	28.43
	30	31.02	28.44	97.39	0.41	0.86	2.69
	40	32.96	22.30	69.74	1.76	2.38	7.24
$B3$	10	17.89	19.80	73.31	6.40	17.92	50.28
	20	12.62	6.08	20.32	0.56	0.70	1.24
	30	10.50	4.68	14.73	0.15	0.29	0.79
	40	13.46	5.31	20.54	0.20	0.37	1.14
	50	10.88	5.77	18.91	0.08	0.10	0.22
$B4$	10	11.40	6.13	21.55	1.43	3.16	10.01
	20	10.83	5.60	17.87	0.55	0.70	1.76
	30	9.66	4.17	14.24	0.14	0.28	0.78
	40	10.87	6.66	16.68	0.49	0.35	0.75
$C3$	10	35.47	31.79	111.33	10.34	15.22	51.56
	20	21.46	14.45	52.05	5.51	8.30	26.83
	30	17.86	11.98	49.27	3.90	6.32	22.45
	40	16.55	6.68	26.89	1.31	1.20	4.43
	50	16.49	7.46	30.21	2.75	3.33	10.74

The accuracy of lower bounds can be further improved, as Table 2 shows, when use is made of the routine sketched in Section 2.5. The column "Score" shows in how many cases out of ten the initial lower bounds was found to be equal to the true quadratic optimum.

Table 2

Version	n	$\frac{OPT - LB}{LB}\%$ Av	Sd	Max	Score
$B3$	10	4.39	12.64	42.29	8
	20	0.23	0.46	1.24	8
	30	0.08	0.24	0.79	9
	40	0.19	0.36	1.14	7
	50	0.04	0.09	0.22	8
$C3$	10	3.17	6.35	16.02	8
	20	0.78	2.00	6.70	8
	30	0.29	0.45	1.07	7
	40	0.47	0.47	1.38	3
	50	0.90	1.82	4.97	8
	60	0.34	0.87	2.94	6

Table 3

Version \ n	10	20	30	40	50	60	70
$B1F$	0.09	1.65	21.28				
$B2$	0.08	0.85	8.78				
$B2F$	0.04	0.40	3.93	37.34			
$B3$	0.09	0.49	5.89	43.99			
$B3F$	0.056	0.23	2.56	14.45	78.83		
$C3$	0.036	0.19	1.54	9.46	27.93	68.92	
$C3F$	0.033	0.17	1.17	7.88	29.22	78.36	318.714
$C3F_3$			1.53	9.31			
$C3P$	0.04	0.21	1.13	8.24	27.78		
$C3F_1P$	0.042	0.25	1.23	9.18	30.30		

Notice that, with this improvement, the lower bounds given by $C3$ become comparable with those given by $B3$.

In Tables 3 and 4, the performance of the above mentioned ten variants[4] of the algorithm is compared in terms of average running time per problem (in CPU seconds on the IBM 370/168) and in terms of average number of linear relaxations generated (upper figure) and solved (lower figure) during the branch-and-bound search.

The averages have been estimated, as usual, from a sample of ten test problems for each size n. It should be remarked that, since forcing a variable x_r to the value α ($\alpha = 0$ or 1) is always followed by a forced branching along the opposite direction $x_r = 1 - \alpha$, (which implies the generation of a new relaxation and thus delays a possible fathoming by bound), the number of linear relaxations *generated* is usually higher in those variants which include forcing routines.

The following remarks can be made:

— The algorithms' performance is heavily conditioned by the particular type of UP chosen.

— The C3 variants unquestionably outperform the other ones. None of the variants $C3$, $C3F$ and $C3F_3$ seems to be definitely dominant over the other ones in terms of running time.

— The forcing routine tends to increase the number of linear relaxations generated and to decrease the number of those solved. While it is very effective in B type variants (with savings of 50% and more) it loses its effectiveness in C type variants, where the times for generating a linear relaxation are roughly comparable with those needed for solving it.

— Use of penalties sensibly reduces the number of nodes explored, but obviously increases the amount of work per node. In the range of problem

[4] Because of its expensiveness, variant $B4$ was only used to solve the initial problem.

Table 4

Version \ n	10	20	30	40	50	60	70
$B1F$	65.4	457.9	2868				
	12.7	77.7	426				
$B2$	18.8	63.4	227.4				
	18.8	62.9	227				
$B2F$	29.4	108.2	453.4	2480			
	7	17.5	66.7	322			
$B3$	23.2	38.6	147.6	533.2			
	20.4	32.3	137	526.8			
$B3F$	10.2	45.6	215.2	991.2	3235.4		
	2.8	6.4	34.5	126.1	361.8		
$C3$	12.12	58	259	1138.6	2094.4	3476.4	
	12.12	53.8	251.8	1130.6	2079.8	3449.1	
$C3F$	10.44	50.6	231	1257.7	3329	5556.8	15611
	2.82	7.6	38.9	181.4	321.3	530.9	1787.8
$C3F_3$			239.5	1121.5			
			56.6	284.4			
$C3P$	7.4	36	110.2	587.4	1397		
	7.3	36	110.2	587.4	1396.9		
$C3F_1P$	8.2	51.6	136.6	785.8	2094.9		
	6.4	29.1	87.9	453.4	885.5		

sizes observed, there seems to be no particular advantage in using them. The variant in which they are used together with constraint pairing is not recommended.

Tables 5 and 6 show, for some variants, the effect of decreasing the density of

Table 5

Density \ Size	$\frac{UB-LB}{LB}$%(Av)					$\frac{OPT-LB}{LB}$%(Av)				
	10	20	30	40	50	10	20	30	40	50
Version $B3$										
100%	17.89	12.62	10.50	13.46	10.88	6.40	0.23	0.08	0.20	0.04
50%	27.88	39.63	27.92	37.50		0	1.13	1.53	1.05	
25%	38.27	52.12	44.28	31.44		0	1.29	5.54	0.22	
5%	9.63	21.39	34.77	39.90	28.97	0	2.92	4.34	4.87	3.43
Version $C3$										
100%	35.47	21.46	17.86	16.55	16.49	10.34	5.51	3.90	1.31	2.75
50%	38.63	43.62	28.37	39.57		1.12	2.02	1.86	1.65	
25%	70.60	53.62	40.53	53.64		7.95	0.58	3.18	3.72	
5%	16.55	23.36	36.44	36.49	28.06		2.05	4.53	3.01	2.49

Table 6

Version *B3F*					
	Size:	10	20	30	40
Density:					
100%		0.056	0.23	2.56	14.45
50%		0.047	0.65	6.38	88.50
25%		0.040	0.54	4.53	74.55
5%		0.026	0.11	0.70	5.40
Version *C3*					
	Size:	10	20	30	40
Density:					
100%		0.036	0.20	1.54	9.47
50%		0.039	0.55	6.81	71.71
25%		0.037	0.52	3.99	90.42
5%		0.021	0.12	0.79	6.61
Version *C3F*					
	Size:	10	20	30	40
Density:					
100%		0.033	0.18	1.17	7.88
50%		0.034	0.40	3.93	43.02
25%		0.029	0.30	2.04	50.92
5%		0.018	0.088	0.45	3.28

Q on the tightness of the bounds (Table 5) and on the average running time per problem (Table 6).

All variants are seen to behave better when the matrix is very dense (100%) or very sparse (5%). Fortunately this is actually the case in most applications.

5. Some theoretical properties of upper planes

5.1. Let S be any subset of R^n and f an arbitrary real function whose domain contains S and which has a maximum in S. For convenience, we shall often denote a point $V \equiv (v_0, v_1, \ldots, v_n)$ in R^{n+1} as (v_0, v). Define

$$S^* \equiv \{V \in R^{n+1}: v_0 + vx \geq f(x), x \in S\}$$

by convention, $\phi^* = R^{n+1}$. The following properties hold:

(a) $S \supseteq T \Rightarrow S^* \subseteq T^*$;

(b) $(S \cup T)^* = S^* \cap T^*$;

(c) $(S \cap T)^* \supseteq S^* \cup T^*$;

(d) S^* is non empty, convex and unbounded;

(e) If S is finite, S^* is a convex polyhedron;

(f) If f is convex and S is a convex polytope (i.e., a bounded convex polyhedron), then $S^* = E(S)^*$, where $E(S)$ is the set of the vertices of S;

(g) If f is convex and S is a convex polytope, S^* is a convex polyhedron.

The proof of properties (a) through (e) is straightforward[5]. (g) follows at once from (e) and (f). Let us prove (f).

By property (a), $E(S)^* \supseteq S^*$. On the other hand, let $V \equiv (v_0, v) \in E(S)^*$. Any $x \in S$ is a convex combination $\lambda_1 z_1 + \cdots + \lambda_s z_s$ of the vertices of S. Hence

$$v_0 + vx = v_0\left(\sum_{i=1}^{s} \lambda_i\right) + v\left(\sum_{i=1}^{s} \lambda_i z_i\right) = \sum_{i=1}^{s} \lambda_i (v_0 + v z_i) \geq \sum_{i=1}^{s} \lambda_i f(z_i)$$
$$\geq f\left(\sum_{i=1}^{s} \lambda_i z_i\right) = f(x).$$

Thus $V \in S^*$ and $E(S)^* = S^*$.

5.2. For simplicity, we shall assume now S to be a compact set. For any $V \in S^*$, $\max_{x \in S}(v_0 + vx)$ does exist and is an upper bound for $z = \max_{x \in S} f(x)$. On the other hand, there is at least one $\tilde{V} \in S^*$ for which such an upper bound actually *equals* z (it suffices to take $\tilde{V} = (z, 0, \ldots, 0)$!). Hence we have

Proposition 5.1.

$$\min_{V \in S^*} \max_{x \in S} (v_0 + vx) = \max_{x \in S} f(x).$$

This equality suggests that the maximum of f in S could be found, in principle, by minimizing over S^* the convex function $h(V) = \max_{x \in S}(v_0 + vx)$. Notice that, if $h(V) = v_0 + v\bar{x}$, then $[1 : \bar{x}]$ is a subgradient of h in V, since, for any $V' \in S^*$, we have

$$h(V') \geq v_0' + v'\bar{x} = h(V) + [1 : \bar{x}](V' - V).$$

Corollary 5.2. *If* $h(\tilde{V}) = \min_{v \in S^*} h(V)$ *and* $f(\bar{x}) = \max_{x \in S} f(x)$, *then* $\tilde{v}_0 + \tilde{v}x$ *attains its maximum over* S *in* $\bar{x}$.

Proof. for all $x \in S$, $\tilde{v}_0 + \tilde{v}\bar{x} \leq h(\tilde{V}) = f(\bar{x}) \leq \tilde{v}_0 + \tilde{v}\bar{x}$.

5.3. Remark. It is worth noting that, if we restrict ourselves to *homogeneous* UPs vx, it is still true that

$$\min\left\{\max_{x \in S} vx : v \in R^n,\ vx \geq f(x) \forall x \in S\right\} \geq \max_{x \in S} f(x), \tag{5.1}$$

but equality does not have to hold, as the following example shows:

[5] Properties (a), (b), (c) are satisfied, more generally, by any *Galois connexion* between the subsets of R^n and the subsets of R^{n+1}.

Example.

$$f(x) = 18x_1x_2 + 2x_1x_4 + 2x_1x_5 + 2x_2x_3 + 2x_2x_6 + 10x_3x_4 + 2x_3x_6 + 2x_4x_5 + 10x_5x_6,$$

$$S \equiv \{x \in B^6: 5x_1 + 4x_2 + 4x_3 + 2x_4 + 3x_5 + 2x_6 \leq 10\}.$$

The maximum of $f(x)$ in S is 18. Equality in (5.1) would imply the existence of a vector v such that

$$18 \leq v_1 + v_2 \leq 18,$$

$$10 \leq v_3 + v_4 \leq 18,$$

$$10 \leq v_5 + v_6 \leq 18,$$

$$6 \leq v_1 + v_4 + v_5 \leq 18,$$

$$6 \leq v_2 + v_3 + v_6 \leq 18,$$

since the points (1, 1, 0, 0, 0, 0), (0, 0, 1, 1, 0, 0), (0, 0, 0, 0, 1, 1), (1, 0, 0, 1, 1, 0), (0, 1, 1, 0, 0, 1) belong to S. But the above system of inequalities is clearly inconsistent.

5.4. Let us now come back to the particular case in which $f(x) \equiv x^TQx$ and $S \equiv X = \{x \in B^n: ax \leq b\}$. Since X is finite, X^* is a convex polyhedron by the above property (e). An explicit description of X^* as intersection of half spaces is in practice very difficult to obtain, because of the usually large number of points in X. However, property (a) above suggests the possibility of getting more tractable subsets of X^* from suitable extensions of X.

Let us consider, for instance, the extension $\check{X} \equiv \{x \in R^n: x \geq 0, ax \leq b\} \supset X$. Define p as the vector whose jth component is $(b/a_j)q_{jj}$.

Theorem 5.3. *If Q is positive semidefinite, px is an UP for x^TQx in $\check{X}$. Moreover,*

$$h(0, p) = \min_{(v_0, v) \in \check{X}^*} h(v_0, v).$$

Proof.

$$\check{X}^* = E(\check{X})^* = \left\{0, \frac{b}{a_1}u_1, \ldots, \frac{b}{a_n}u_n\right\}^*$$

$$= \left\{(v_0, v): v_0 \geq 0, v_j + \frac{a_j}{b}v_0 \geq \frac{b}{a_j}q_{jj} \text{ for } j = 1, \ldots, n\right\}.$$

From the last expression of $\check{X}^*$ it is apparent that the point (0, p) belongs to

$\check{X}^*$. On the other hand, for all $(v_0, v) \in \check{X}^*$ and $x \in \check{X}$, one has

$$v_0 + \sum_{j=1}^{n} v_j x_j \geq v_0 + \sum_{j=1}^{n} \left(\frac{b}{a_j} q_{ij} - \frac{a_i}{b} v_0 \right) x_j = v_0 \left(1 - \frac{\sum_{j=1}^{n} a_j x_j}{b} \right) + \sum_{j=1}^{n} p_j x_j \geq \sum_{j=1}^{n} p_j x_j.$$

This proves the second half of the theorem.

The upper plane px has been tested on randomly generated test problems ranging from 10 to 80 variables. The results, except for some very tight problems (i.e., with $b \leq 0.2 \sum_{j=1}^{n} a_j$), showed a poor behaviour, due perhaps to the fact that in most observed cases there were a few p_j's taking large values, which have sharply pushed up the optimum value of the linear relaxation, in spite of the fact that the vast majority of the p's were low.

A slight improvement was obtained by replacing the above considered $\check{X}$ with $\check{X}_r \equiv \{x \in R^n\colon x \geq 0, ax \leq b, x_r \leq 1\}$, where $p_r = \max_{1 \leq j \leq n} p_j$. It is hoped that tighter extensions of X could lead to significant improvements.

Acknowledgement

Acknowledgements are expressed for the partial support given by the Istituoto per le Applicazioni del Calcolo (Rome) and the Centro Nazionale Universitario di Calcolo Elettronico (Pisa) of the Consiglio Nazionale delle Recherche, as well as the National Research Council of Canada (Grant A8552) and Nato's System Science Exchange Grant.

References

[1] M.L. Balinski, "Fixed cost transportation problems", *Naval Research Logistics Quarterly* 11 (1961) 41–54.

[2] M.L. Balinski, "On a selection problem", *Management Science* 17 (3) (1970) 230–231.

[3] E.M.L. Beale, "An algorithm for solving the transportation problem when the shipping cost over each route is convex", *Naval Research Logistics Quarterly* 6 (1959) 43–56.

[4] V.A. Cabot and R. Francis, "Solving certain non-convex quadratic minimization problems by ranking the extreme points", *Operations Research* 18 (1970) 82–86.

[5] P.L. Hammer and P. Hansen "Quadratic 0–1 programming", discussion paper No. 7129, CORE. (Heverlee, 1972).

[6] P.L. Hammer, M. Padberg and U.N. Peled, "Constraint pairing in integer programming", *INFOR. Canadian Journal of Operational Research and Information Processing* 13 (1975) 68–81.

[7] R.M. Karp, "Reducibility among combinatorial problems", technical report 3, Computer Science University of California (Berkeley, CA, 1972).

[8] A. Land, personal communication (1975).

[9] D.J. Laughhunn, "Quadratic binary programming with applications to capital budgeting problems", *Operations Research* 18 (1970) 454–461.

[10] C.C. Petersen, "A capital budgeting heuristic algorithm using exchange operations", *AIIE Transactions* 6 (1974) 143–150.

[11] J. Rhys, "A selection problem of shared fixed costs and network flows", *Management Science* 17 (3) (1970) 200–207.
[12] H. Salkin and C.A. de Kluyver, "The knapsack problem: a survey", *Naval Research Logistics Quarterly* 22 (1) (1975) 127–144.
[13] H.A. Taha, "Concave minimization over a convex polyhedron", *Naval Research Logistics Quarterly* 20 (1973) 533–548.
[14] C. Witzgall, "Mathematical methods of site selection for Electronic Message Systems (EMS)", NBS internal report (1975).