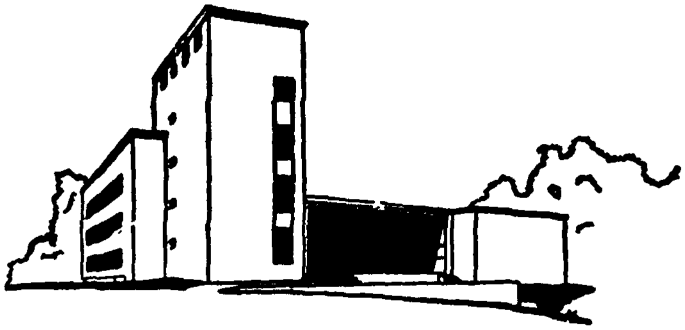
Carnegie-Mellon University

PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION

WILLIAM LARIMER MELLON, FOUNDER



30

W. P. 41-71-2

Management Sciences Research Report No. 272

GENERAL QUADRATIC PROGRAMMING

by

Claude-Alain Burdet

November 1971

This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie-Mellon University, under Contract N00014-67-A-0314-0007 NR 047-048 with the U. S. Office of Naval Research. Reproduction in while or in part is permitted for any purpose of the U. S. Government.

Management Sciences Research Group
Graduate School of Industrial Administration
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

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) Graduate School of Industrial Administration Carnegie-Mellon University	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP Not applicable

3. REPORT TITLE

GENERAL QUADRATIC PROGRAMMING

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Management Sciences Research Report November 1971

5. AUTHOR(S) *(First name, middle initial, last name)*

Claude-Alain Burdet

6. REPORT DATE November 1971	7a. TOTAL NO. OF PAGES 27	7b. NO. OF REFS 14
8a. CONTRACT OR GRANT NO. N00014-67-A-0314-0007 b. PROJECT NO. NR 047-048 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) Management Sciences Research Report No. 272 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* W.P. 41-71-2	

10. DISTRIBUTION STATEMENT

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

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Logistics and Mathematical Statistics Br. Office of Naval Research Washington, D. C. 20360

13. ABSTRACT

An algorithm is presented for the general (not necessarily convex or concave) quadratic programming problem over a linearly constrained set. The algorithm is finitely convergent and makes use of a convex quadratic programming method as a subroutine (like the quadratic simplex for instance). The basic tool for this method is a facial decomposition for polyhedral sets.

Unclassified

Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
non-convex programming						
quadratic programming						
negative-positive definite, semi-definite, indefinite quadrics						
facial decomposition						
finite algorithm						

ABSTRACT

An algorithm is presented for the general (not necessarily convex or concave) quadratic programming problem over a linearly constrained set. The algorithm is finitely convergent and makes use of a convex quadratic programming method as a subroutine (like the quadratic simplex for instance). The basic tool for this method is a facial decomposition for polyhedral sets.

Section 1) Introductory Remarks

The term quadratic programming has now become classical both in management and engineering sciences; it refers to the optimization of a quadratic objective function over a polyhedral set P in the n-dimensional vector space . Usually, however, all that is meant is convex quadratic programming, i.e. solving problems of either type:

a) minimize $f(x)$, subject to $x \in P$ (1a)

or b) maximize $g(x)$, subject to $x \in P$ (1b)

where f is a convex function ,

g a concave function ,

and P a polyhedral set defined by say $A x \leq b$ with A an m by n matrix.

Convex quadratic programs are indeed very important because they appear in practical applications and also because they represent a useful approximation of more general convex programs.

There are other cases of linearly constrained quadratic programs, however,(which also correspond to practical problems) namely the opposite of (1) called here concave programming (quadratic) problems, i.e.

a) maximize $f(x)$, subject to $x \in P$ (2a)

or b) minimize $g(x)$, subject to $x \in P$ (2b)

A third class of quadratic program is the general one (as it contains both the convex and concave cases), where one seeks the optimum (maximum and/or minimum) of a general quadratic function $f(x)$ (not necessarily convex or concave) over the polyhedral set P .

This paper focuses on this general case, presenting an algorithmic principle based on some simple fundamental properties of Γ. Several versions of such algorithms are being tested in order to find appropriate measures of efficiency for algorithms of this type.

Probably because it does not possess the many useful properties of its convex special case, relatively little attention has been devoted to

Footnote

the general case in the literature.[1] With no attempt to present a complete bibliography of the subject, let us mention the articles [9 , 10] by Ritter who is apparently the first to have studied the non-convex case, followed by Cottle and Mylander [5] and more recently Konno [7]. All

Footnote

of these papers[2] are centered around the concept of cutting planes and [7] presents an interesting study of several types of cutting planes applicable to non-convex quadratic programming, including the cuts introduced by Hoang Tui in [11] for general concave programming and which have recently received much attention in integer programming [1,3,6,8].

Our approach proceeds along different lines, however, as it is fundamentally oriented towards an enumerative decomposition into subproblems

Footnote

of the original problem.[3] It makes use of the so-called facial decomposition of the polyhedral set P , which leads to the construction of an arborescence along which one may search for the global optimum; much as for the branch and bound algorithms, the arborescence is used here to isolate "candidates" (usually local optima) for the optimum which then can be enumerated to identify the over-all best solution(s), i.e. the global optimum(s).

In section 2 we list some useful properties of the linearly constrained general quadratic programming problems. Section 3 then contains a brief summary of the facial decomposition procedure [4] and section 4 outlines some relevant mathematical developments.

Section 2) <u>The structure of quadratic programs</u>

We present below a list of the properties which lie at the crux of the problem under consideration; all are straightforward observations and do not require much mathematical insight. For simplicity we only consider a bounded n-dimensional polyhedron P as feasible region and the general quadratic problem (GQP) is defined by:

$$\text{minimize } f(x)\ ,\ \text{subject to } x \in P$$

where f is a quadratic function.

Call S the set of all optimal solutions $\bar{x}$ of GQP .

<u>Theorem 1</u>: *If $S \subset$ rel int (P) then f is strictly convex* (i.e., positive definite) on Aff(P) and S consists of a unique point $\bar{x}$.

Footnote <u>Corollary 1.1</u>:[4/] If f is not strictly convex on P then f attains its global minimum on the relative boundary of P .

<u>Theorem 2</u>: The relative boundary of an n-dimensional polyhedron P is the union of its (n-1)-dimensional (closed) facets.

<u>Theorem 3</u>: If f is convex (concave) on P then it is convex (concave) on all the facets, and lower-dimensional faces F of P .

<u>Corollary 3.1</u>: Since all faces of P are polyhedral, theorem 3 also holds for any face of P.

<u>Theorem 4</u>: If $x \in P$ then x is either a vertex of P or x belongs to the relative interior of some face of P (including P itself).

Corollary 4.1: Any optimal solution $\bar{x}$ of GQP satisfies theorem 4.

Theorem 5: Let $F_2 \subset F_1 \subset P$ be faces of P, and let $\bar{x} \in F_1$ be optimal, i.e.

$$f(\bar{x}) \leq f(x) \;, \quad \forall\, x \in F_1 \;.$$

If $\bar{x} \in \text{rel int } (F_1)$

then $f(\bar{x}) < f(x) \;, \quad \forall\, x \in F_2$

Theorem 6: If f is concave on a face $F \subset P$ then there exists a vertex $\bar{x}$ of F such that

$$f(\bar{x}) \leq f(x) \;, \quad \forall\, x \in F$$

Proofs: To 1: Along any 1-dimensional line, the quadratic function f is either strictly concave, strictly convex or linear. Hence, for every point $\bar{x} \in S$ and along any line $L(\bar{x})$ through $\bar{x}$, f must be convex or linear since $f(\bar{x}) \leq f(x) \;, \; \forall\, x \in L(\bar{x})$; but suppose that f is a linear function along every line $L(\bar{x})$ through $\bar{x}$; then on the relative boundary of P, there must exist a minimal point $\bar{\bar{x}} \in (L(\bar{x}) \cap S)$, and we have a contradiction to the hypothesis; hence, f is strictly convex; furthermore, a strictly convex function attains a unique minimum $\bar{x}$ on any open set $U(S)$ S for instance $U = \text{rel int}(P)$; hence, $S = \{\bar{x}\}$. Q.E.D.

To 1.1: By contraposition of theorem 1.

To 2: This is a classical result and the proof is omitted.

To 3: Immediate since $F \subset P$.

<u>To 4</u>: This is a classical result, too. The polyhedron P can be decomposed into disjoint subsets $\varphi_\nu = \text{rel int } (F_\nu)$ where F_ν runs over all faces of P (including P itself);furthermore, since the relative interior of a point is ill-defined, one has

$$P = \bigcup_\nu \varphi_\nu \cup \{F_\mu\}$$

where F_μ are the vertices of P and F_ν the k-dimensional faces of P, $(1 \leq k \leq n)$.

<u>To 4.1</u>: Immediate since $\bar{x} \in P$.

<u>To 5</u>: Immediate since $F_2 \subset F_1$ but $F_2 \not\subset \text{rel int } (F_1)$.

<u>To 6</u>: Proof omitted.

The above properties of GQP lead in a straightforward manner to the algorithmic principle presented at the end of section 2.

In theorem 4 (its proof and corollary) one finds the germ of a construction described below in greater detail.

Section 3) The faces of a polyhedral set P .

Consider the system

$$Ax = b \tag{3a}$$

$$x \geq 0 \tag{3b}$$

with A an m by $n + m$ matrix of rank m,

and set

$$M = \{1,2,\dots,n,n+1,\dots,n+m\} ;$$

A subset $I \subset M$ is called minimal if

$$\{x | x_i \geq 0, \forall i \in I\} \subset \{x | x_i \geq 0, \forall i \in M\} = P$$

and for every $i_o \in I$, there exists a point $\bar{x}$ such that

$$\bar{x}_{i_0} < 0, \text{ while } \bar{x}_i \geq 0, \forall i \in I - \{i_o\}.$$

Geometrically, it may be seen that the minimal set I consists of the indices $i \in I$ for which the affine set $\{x | x_i = 0, i \in I\}$ contains at least one facet of the polyhedral set P defined by (3). Let I_o be the minimal set of P.

Note that a minimal set I_o can be found by solving iteratively the following L.P. starting with $I = M$:

$$\text{minimize } x_{i_o} \text{, subject to } \quad x_i \geq 0 \text{ , } \forall i \in I - \{i_o\} \qquad Ax = b \tag{3c}$$

If the minimal value x_{i_o} is < 0 then keep i_o in the set I; otherwise, if $x_{i_o} \geq 0$, then reduce the set I by eliminating its element i_o.
It is shown in [4] that after all elements $i_o \in M$ have been considered in (3c), the remaining set $I = I_0$ is minimal.

Now the facial decomposition of an n-dimensional polyhedron P simply consist in determing the minimal index set $I = I(i_1)$ corresponding to each (n-1)-dimensional facet $F(i_1) = \{x \in P | x_{i_1} = 0, i_1 \in I_o\}$, the set $I = I(i_1, i_2)$ for each (n-2)-dimensional face

$$F(i_1,i_2) = \{x \in F(i_1) \subset P \mid x_{i_1} = x_{i_2} = 0 \ , \ i_2 \in I(i_1) \subset I_o\} \ ,$$

etc....; each index set $I(i_1,\ldots,i_q)$ characterizes a face $F(i_1,\ldots,i_q)$:

$$F(i_1,\ldots i_q) = \left\{x \in P \mid x_{i_1} = \ldots = x_{i_q} = 0 \ , \quad Ax = b, \quad x_i \geq 0 \ , \ \forall i \in I(i_1,\ldots i_{q-1}) \right.$$

which lies in the subspace $\text{Aff}\big(F(i_1,\ldots,i_q)\big)$ defined by

$$x_{i_1} = x_{i_2} = \ldots = x_{i_q} = 0 \quad ;$$

and, of course, one has

$$F(i_1) \supset F(i_1,i_2) \supset \ldots \supset F(i_1,\ldots,i_q) \ ,$$

$$I(i_1) \supset I(i_1,i_2) \supset \ldots \supset I(i_1,\ldots,i_q) \ ,$$

and $\text{Aff}\big(F(i_1)\big) \supset \text{Aff}\big(F(i_1,i_2)\big) \supset \ \ldots . \ \supset \text{Aff}\big(F(i_1,\ldots,i_q)\big)$.

The complete facial decomposition of P therefore assumes a tree-like structure beginning with a single node which corresponds to the largest face P (which contains all the other faces); from this node, one has branches (as many as there are elements in I_o), each one leading to a facet $F(i)$, $i \in I_o$: then, on the third level, one finds the (n-2)-dimensional faces of P , i.e. the facets of $F(i)$, $i \in I_o$; from each $F(i_1)$, $i_1 \in I_o$ one therefore has branches going to the faces $F(i_1,i_2)$ with $i_2 \in I(i_1) \subset I_o$; and so on...the lowest level containing all the 0-dimensional faces (vertices) of P .

Note that this tree-structure is redundant because $F(i_1,\ldots i_q)$ corresponds to one and the same face for all permutations of the indices

$i_1,\dots,i_q$. In order to avoid this type of redundancy, it suffices to generate the index sets $\{i_1,\dots,i_q\}$ in a strictly increasing lexicographic order. A more detailed description of the facial decomposition method can be found in [4]. In particular, a method is given for eliminating, in the facial arborescence, another kind of redundancy due to degeneracy. Ultimately, the nodes of the arborescence will be in one-to-one correspondence with the faces of P.

Algorithmic principle: The face decomposition of P furnishes, in quite a natural way, an enumerative method to solve linearly constrained quadratic problems. Consider a face F of P generated at some level of the face decomposition:

A) If f is convex on F, then we know that it is convex on every face and subface of F (Theorem 3). We may therefore solve the convex quadratic problem:

minimize $f(x)$, subject to $x \in F$;

then store the optimal solution $\bar{x}(F)$, remembering that it is a candidate for the global optimum of the original problem: minimize $f(x)$, subject to $x \in P$.

Furthermore, since $\bar{x}(F)$ is the optimum on F, there is no need to investigate the faces and subfaces of F (Theorem 3) individually: thus the branch can be terminated at F.

B) If f is concave on F, then we know that the optimum of the concave quadratic problem:

minimize $f(x)$, subject to $x \in F$

is attained on (at least) one vertex of F (Theorem 6); one therefore may proceed with the face decomposition of F, without checking the concavity of f on each face and subface of F, until the lowest level (the vertices) is reached. The determination of the optimal vertex $\bar{x}(F)$ can be made by (explicit or implicit) search in this set of vertices: again, $\bar{x}(F)$ is stored as a candidate for the global optimum of the original problem. Note that one may also use here any other concave quadratic programming algorithm, such as the ones studied in [2] and [7] for instance.

C) The third remaining case is that where f is neither concave nor convex on F; here one simply proceeds with the facial decomposition of F, generating new faces which must be tested for convex- or concavity.

Termination: It is easily seen that an algorithm based on A), B), and C) will terminate in a finite number of steps; indeed the face decomposition generates a finite number of polyhedral sets (faces). Thus the only point which could lead to an infinite sequence of numerical operations is A); however, algorithms for convex quadratic problems on a polyhedral set F (like the quadratic simplex algorithm) are convergent in a finite number of steps.

Finally, one compares all the candidates $\bar{x}(F)$ obtained at the termination of all branches and selects the "best" one(s) as the global optimum(s).

Concluding Remarks:

The above algorithmic principle is not exempt of difficulties inherent to the nature itself of the general quadratic problem over a polyhedral set P :

1) It may happen that f is "essentially concave" on P, that is, that there are only relatively few faces F of dimension greater than zero (i.e., other than vertices) where f is convex. In this case, the efficiency of the algorithm is limited by the same phenomenon as for the concave problems, namely the large number of faces and especially vertices of P (this number grows exponentially with the dimension of P, in general). In order to avoid the explicit enumeration of all the vertices of P, one has to use lower bound (for a minimization problem) estimates in order to truncate branches which are obviously suboptimal; this approach corresponds to the classical branch and bound method. Its efficiency critically depends on the ability of the bound estimates to truncate relatively high -dimensional faces, while requiring only a non-prohibitively large amount of computations to obtain the bounding value. The developments here are highly heuristic and problem-dependent and therefore, outside the scope of this paper.

2) Another disagreeable phenomenon is due to degeneracy in the polyhedral set P. Indeed it may happen that the face decomposition is geometrically redundant, while its

algebraic characterization by index sets $\{i_1,\ldots,i_q\}$ is not. To clarify this point, let us simply mention that two faces F_1 and F_2 with non-identical index sets, i.e.

$$F_1(i_1,\ldots,i_q) \quad , \quad F_2(j_1,\ldots,j_r)$$

with $\{i_1,\ldots,i_q\} \neq \{j_1,\ldots,j_r\}$

may be identical,from a practical point of view,in that their point sets are the same, i.e.

$$\{x \mid x \in F_1\} = \{x \mid x \in F_2\} .$$

Degeneracy can be observed in the construction of the minimal sets I , however, and may therefore be eliminated from the tree-structure by appropriate bookkeeping. Since this is not immediately pertinent to the quadratic programming aspect but rather stems intrinsically from the face decomposition of P , the interested reader is referred to [4] for further details.

3) For the case where the dimension of P is not prohibitively large (say less than 50 to 100) and where the <u>non-convexity</u> of f is not dominant, i.e. where f is convex on all (but a few) low-dimensional faces, the present approach seems promising particularly in view of the inherent difficulty of this type of problem. In any case, it is not difficult to find problems where the approach presented here is clearly superior to cutting plane techniques, simply because it does not entail the convergence obstacles introduced by the latter.

4) Relation to other work.

A similar approach to the facial decomposition has been sketched by Murty in [13]. He shows that the optimal solution can be found in a finite list of optimal solutions to convex quadratic subproblems. The list of subproblems is obtained from a full combinatorial arborescence of affine sets, defined by

$$x_{i_1} = x_{i_2} = \dots = x_{i_k} = 0 \ , \quad i \in M$$

where $$0 \leq k \leq n \ ;$$

Finiteness of the procedure follows because:

a) There are at most $\Delta = \sum_{k=o}^{n} C^k_{n+m} \leq 2^{n+m}$ distinct affine sets in the arborescence;

b) Only the reduced problems which are of the convex quadratic type need be solved (in finitely many steps).

In Murty's procedure $m_1 = m + (n - k)$ new subproblems are generated in the arborescence for every affine set $S(n_1, m_1)$ of dimension $n_1 = (n - k)$ with properties

(ii) $S(n_1, m_1)$ intersects the feasible set,

(iii) The objective function f is not convex on $S(n_1, m_1)$

(Note that if no additional bookkeeping organization is implemented this procedure generates the same affine set $k!$ times, corresponding to the permutations of $i_1,\dots,i_k$)

The facial decomposition algorithm presented in section 4 generates distinct subproblems and their enumeration is curtailed according to the remarks below

1) Only faces of the feasible set P, i.e. only those affine sets $S(n_1,m_1)$ which are known a priori to intersect P, are generated.

2) Because the convexity tests indicate that f is either convex, or concave, or neither convex nor concave on the particular face S under consideration, there is no extra cost for using this information (when the function to be minimized (maximized) on S turns out to be concave (convex)). Indeed, when f is concave on S, we know a priori that the vertices of S can be generated directly with no additional convexity test (see part B) of the algorithmic principle and STEP 4 B of the algorithm).

Finally let us note that, in general, the above remarks 1) and 2) may be expected to generate, for the amount of computations of the facial decomposition method, a reduction which grows exponentially with the dimension n of the problem as compared to the approach of [13].

4) <u>Computational aspects</u>.

Consider the set of linear constraints

$$A' x' \leq b$$

where $$x' \geq o$$

A' is an m by n matrix

x' an n-vector $(x_1,\dots,x_n)$

b an m-vector $(b_{n+1},\dots,b_{n+m})$

As customary in linear programming, let us introduce slack variables x'_k

$$x'_k = b_k - \sum_{j=1}^{n} a'_{kj} \; x'_j \qquad , \quad k = n+1,\dots,n+m$$

System (3) then becomes

$$Ax = b \tag{4a}$$

$$x \geq o \tag{4b}$$

with $A = (A', I_m)$ where I is a m by m unit matrix,

and $x = (x_1,\dots,x_n,x_{n+1},\dots,x_{n+m})$.

Furthermore, take the objective function $f(x)$ to be defined by

$$f(x') = x'^T Cx' + Dx' + f^o \tag{5}$$

where C is a n by n <u>symmetric</u> matrix

D an n-vector

f^o a scalar.

(This is the general form for a quadratic function in n variables x' .)

Substituting into a quadratic form:

A face F_J of P characterizes a set of variables x_j, $j \in J$ which are kept at value zero

$$x_j = o \qquad \forall j \in J \subset \{1,2,\ldots,n,n+1,\ldots,n+m\} = M \tag{6}$$

and we need represent the quadratic function f in the affine space $\mathrm{Aff}(F_J)$ defined by (6). Consider a basis B for the system (4), where the set J is contained in the non-basic set. The general solution to (4a) can be described by

$$x_B = B^{-1}b - B^{-1}Nx_N \tag{7}$$

where B is a set of m linearly independent colums of A, forming a non-singular m by m matrix (basis);
and N contains the remaining columns of A; the variables x_B are called "basic" and x_N "non-basic."

A general point in the affine space $\mathrm{Aff}\ (F_J)$ defined by (6) then reads

$$x_B = B^{-1}b - B^{-1}N_J\, x_{N_J} \tag{8a}$$

$$x_{N_J} \quad \text{arbitrary} \tag{8b}$$

$$x_J = o \tag{8c}$$

where the matrix N_J is obtained from the submatrix N by deleting the columns corresponding to $j \in J$.

Let us substitute (8) into (5) in order to obtain the expression for the function f, restricted to a smaller domain of definition, namely $\mathrm{Aff}\ (F_J)$.

Since only the original variables $x' = (x_1, x_2, \ldots, x_n)$ are present in the argument of the function f, we only need substitute for the variables x'_j, $j \in \{1,2,\ldots,n\}$ and basic, i.e.

$$x'_B = (B^{-1})'b - (B^{-1})'N_J\, x_{N_J} \tag{9}$$

$$= \beta b - \beta N_J\, x_{N_J}$$, where β denotes the matrix $(B^{-1})'$ obtained from (B^{-1}) by deleting the rows corresponding to basic variables x_j $j \in \{n+1,\ldots,n+m\}$.

Let us now substitute (9) into (5), yielding

$$f = \left[\beta b - \beta N_J x_{N_J}\right]^T C \left[\beta b - \beta N_J x_{N_J}\right] +$$

$$+ D\left[\beta b - \beta N_J x_{N_J}\right] + f^o =$$

$$= (f^o + b^T\beta^T C\beta b + D\beta b) +$$

$$+ (-D\beta N_J x_{N_J} - b^T\beta^T C\beta N_J x_{N_J} - x^T_{N_J} N^T_J \beta^T C\beta b)$$

$$+ x^T_{N_J} N^T_J \beta^T C\beta N_J x_{N_J}$$

$$= x^T_{N_J} N^T_J C_J N_J x_{N_J} + D_J N_J x_{N_J} + f^o_J \tag{10a}$$

where

$$C_J = \beta^T C \beta \text{ , which is (again) a \underline{symmetric} matrix} \tag{10b}$$

$$D_J = (D\beta + 2\, b^T C_J) \tag{10c}$$

$$f^o_J = f^o + D_J b + b^T C_J b \tag{10d}$$

Thus, one obtains for f, restricted to Aff (F_J), a function $f_J(x_{N_J})$ which is again a (general) quadratic function.

Testing the convexity of a quadratic form.

It was indicated in section 3 that one of the major ingredients of the algorithmic principle developed there is the test for convexity (and concavity) of the given quadratic objective function on a subset F_J of its domain of definition. To be precise, we must repeatedly explore the definiteness of $f_J(x)$, for $x \in F_J \subset P$, i.e. on a face F_J of P.

According to the above result (10), this task can be seen to consist in finding the definiteness of the quadratic form in the $(n-J)$ variables x_{N_J}:

$$f_J(x_{N_J}) = f_{J_o} + D_J N_J x_{N_J} + x_{N_J}^T N_J^T C_J N_J x_{N_J} \tag{10}$$

One has:

> f_J is convex if the symmetric matrix $N_J^T C_J N_J$ is positive (semi-) definite;

hence, it is sufficient to investigate the definiteness of the symmetric matrix

$$\tilde{C} = N_J^T \beta^T C \beta N_J \tag{10a}$$

There are many theoretical and practical approaches, concievable at this point and we present but one method below, chosen because it seems adequate:

Consider the quadratic function

$$g(\tilde{x}) = \tilde{x}^T \tilde{C} \tilde{x} \tag{11}$$

where $\tilde{C}$ is a symmetric k by k matrix; setting

$$y = \tilde{C} \tilde{x}\ , \quad \text{or} \quad y_i = \sum_{j=1}^{k} \tilde{c}_{ij} x_j \quad , \quad i = 1,2,\ldots, k \tag{12}$$

$g(\tilde{x})$ becomes the scalar product $\tilde{x}y$ of the k-vectors $\tilde{x}$ and y :

$$g(\tilde{x}) = \tilde{x}y = \sum_{j=1}^{k} \tilde{x}_j\, y_j \ .$$

Now, by pivoting in a non-zero diagonal element of $\tilde{C}$ (say $\tilde{c}_{11}$) one can express a variable $\tilde{x}_j$ (here $\tilde{x}_1$) in terms of the $(k-1)$ remaining $\tilde{x}$ variables and one y variable (here y_1) ; this operation is known as a gaussian exchange(principal pivoting)in the linear system (12).

More concretely, one has after the gaussian exchange with pivot $\tilde{c}_{11}$ $(\neq 0)$

$$-\tilde{x}_1 = \frac{1}{\tilde{c}_{11}} \left(-y_1 + \sum_{j=2}^{k} \tilde{c}_{1j}\tilde{x}_j\right) = \frac{-1}{\tilde{c}_{11}}\, y_1 + \sum_{j=2}^{k} \frac{\tilde{c}_{1j}}{\tilde{c}_{11}}\, \tilde{x}_j \tag{13a}$$

and for $i = 2,\ldots,k: y_i = \sum_{j=2}^{k} \tilde{c}_{ij}\tilde{x}_j + \tilde{c}_{i1}\tilde{x}_1 = \sum_{j=2}^{k} \tilde{c}_{ij}\tilde{x}_j - \frac{\tilde{c}_{i1}}{\tilde{c}_{11}} \left(-y_1 + \sum_{j=2}^{k} \tilde{c}_{1j}\tilde{x}_j\right)$

$$= \frac{\tilde{c}_{i1}}{\tilde{c}_{11}}\, y_1 + \sum_{j=2}^{k} \left(\tilde{c}_{ij} - \frac{\tilde{c}_{i1}\,\tilde{c}_{1j}}{\tilde{c}_{11}}\right) \tilde{x}_j \tag{13b}$$

Let us now replace $\tilde{x}_1$ and y_i by their respective expressions (13) in $g(\tilde{x})$,

$$g = \tilde{x}_1 y_1 + \sum_{s=2}^{k} \tilde{x}_s y_s = -\left(\frac{-1}{\tilde{c}_{11}} y_1 + \sum_{j=2}^{k} \frac{\tilde{c}_{1j}}{\tilde{c}_{11}} \tilde{x}_j\right) y_1 + \sum_{s=2}^{k} \tilde{x}_s \left[\frac{-\tilde{c}_{s1}}{\tilde{c}_{11}} y_1 + \sum_{j=2}^{k} \left(\tilde{c}_{js} - \frac{\tilde{c}_{s1}\tilde{c}_{1j}}{\tilde{c}_{11}}\right) \tilde{x}_j\right]$$

$$= \frac{1}{\tilde{c}_{11}} y_1^2 + y_1 \left[- \sum_{j=2}^{k} \frac{\tilde{c}_{1j}}{\tilde{c}_{11}} \tilde{x}_j + \sum_{s=2}^{k} \frac{\tilde{c}_{s1}}{\tilde{c}_{11}} \tilde{x}_s\right] + \sum_{j=2}^{k} \left(\tilde{c}_{js} - \frac{\tilde{c}_{s1}\tilde{c}_{1j}}{\tilde{c}_{11}}\right) \tilde{x}_s \tilde{x}_j$$

$$= \frac{1}{\tilde{c}_{11}} y_1^2 + \sum_{j=2}^{k} \bar{c}_{js} \tilde{x}_s \tilde{x}_j \tag{14}$$

since $\tilde{c}_{1j} = \tilde{c}_{s1}$ for $j = s$ by symmetry of $\tilde{C}$.

Furthermore, by construction, the matrix $\bar{C}$ is again symmetric:

$$\bar{c}_{js} = \tilde{c}_{js} - \frac{\tilde{c}_{s1}\tilde{c}_{1j}}{\tilde{c}_{11}} = \tilde{c}_{sj} - \frac{\tilde{c}_{j1}\tilde{c}_{1s}}{\tilde{c}_{11}} = \bar{c}_{sj} \tag{15}$$

Now since $y_1^2 \geq 0$, one has the necessary condition:

$$g \text{ positive definite} \Rightarrow \tilde{c}_{11} > 0 ,$$

and similarly

$$g \text{ negative definite} \Rightarrow \tilde{c}_{11} < 0 \ .$$

The test for <u>strict convexity of the given (general) quadratic function</u> f , <u>on a restricted domain of definition</u> (Aff F_J) can be made as follows:

<u>Step 1</u>: Check that all elements c_{ii} of C_J in the diagonal are <u>positive</u>; if this is not the case then g is <u>not strictly convex</u>; <u>stop</u>

<u>Step 2</u>: Pivot on c_{11} and generate the new symmetric matrix C_J ; note that the number of rows and columns of $\bar{C}$ is <u>one less</u> than for C_J ; go to Step 1.

In at most (n-J) iterations this procedure shows that C_J is positive definite or not.

<u>Remarks</u>: Since the gaussian elimination is always applied to a symmetric matrix, yielding a new matrix which is again symmetric, explicit computations need only be carried in the upper triangular part of C ; this represents a substantial reduction in the number of necessary operations. In fact principal pivoting is a standard manipulation in mathematical programming, which can be done efficiently.

Note that in Step 2, the choice of c_{11} was arbitrary; in fact an efficient algorithm will try to exhibit non-convexity as soon as possible; thus one will choose the index i of the principal pivot c_{ii} , with greater care.

Outline of the algorithm:

We may now summarize the various results obtained in the previous sections into an algorithm to solve general quadratic, linearly constrained problems. The algorithm described below is but one version of the principle presented at the end of section 3. Since numerical computations are still at an experimental stage, it is difficult to assess the computational efficiency of one version with respect to another, especially when the differences stem from programming details rather than basically different approaches. The algorithm below was chosen because of its relative simplicity of exposition combined with a robust numerical performance.

The algorithm requires the following quantities:

two $(n+1)$-arrays : m , m_I $[0:n]$ and $(n+1)$ "dynamic" arrays (at most $(n+m)$ dimensional) : $I[o;\ ]$, $I[1;\ ]$,...,$I[n;\ ]$.

The result is found in the $(n+m)$-array x_{OPT} with the optimal value for f denoted by OPT .

STEP 0:

Let t : iteration index (level in the arborescence)

m : pointer

I : minimal index set

m_I: number of elements in I

OPT: currentoptimal value for f over P .

initialization: $t := 1$; $m[0] := 1$; $I[0;\ ] := I_o$ = minimal set of P;

$m_I\ [0]$ = number of elements in I_o ; OPT $= +\infty$ (minimization)

<u>STEP 1</u>:

- Test the definiteness of f on the affine set: $\mathrm{Aff}(F_{t-1})$:

  $\mathrm{Aff}\ (F_{t-1}) = \{x \mid x_j = 0\ ,\ \text{for all}\ j = I[0;m[0]],\ldots,I[t-1;m[t-1]]\}$;

- If f is <u>convex</u> then go to STEP 4a ;
- If f is <u>concave</u> then go to STEP 4b ;

<u>STEP 2</u>:

- Find the minimal index set $I[t;\ ]$ of the polyhedral set F_{t-1}

$$F_{t-1}\ :\ \begin{cases} (A,I)\ x = b \\ x \geq 0 \\ x_j = 0\ ,\ \text{for all}\ j = I[0;m[0]],\ldots \\ \qquad\qquad\ldots,I[t-1;m[t-1]] \end{cases}$$

- Denote by $m_I[t]$ the number of elements in $I[t;\ ]$;

<u>STEP 3</u>:

- Let $m[t] := 1$;

  $t := t + 1$;

- go to STEP 1 ;

<u>STEP 4a</u>:

- Solve the <u>convex</u> quadratic problem:

  minimize $f(x)$, subject to $x \in F_{t-1}$

  (using one of the quadratic simplex algorithms, for instance).

- go to STEP 5

<u>STEP 4b</u>:

- Solve the <u>concave</u> quadratic problem

$$\text{minimize} \quad f(x) \text{ , subject to } \quad x \in F_{t-1}$$

(This can be done by determination of the vertices of F_{t-1}, for instance; these vertices are obtained by iteratively computing STEP 2 and STEP 3, bypassing STEP 1, until $t = n$ where the 0-dimensional faces are generated [4]).

<u>STEP 5</u>:

- Denote by $\bar{f}$ and $\bar{x}$ the optimal solution obtained in STEP 4.
- If $\bar{f} \leq OPT$ then set $x_{OPT} := \bar{x}$

  $OPT := \bar{f}$

<u>STEP 6</u>:

- If $m[t-1] \leq m_I[t-1]$ then

  set $m[t-1] := m[t-1] + 1$ and go to STEP 1.

  Otherwise
- set $t := t-1$;
- if $t > 0$ then go to STEP 6
- if $t = 0$ then $\boxed{\text{STOP}}$: the global optimum is x_{OPT}, with value OPT .

The algorithm can be seen to contain the following components:

<u>Branch-identification phase</u> (STEP 2) : Given a $(n-t)$-dimensional face F_{t-1} of the n-dimensional polyhedron P , the minimal set $I[t;]$ is constructed, which identifies all the $(n-t-1)$-dimensional facets of F_{t-1} .

<u>Convexity test phase</u> (STEP 1) : Given a subspace Aff (F_{t-1}) one calculates the restricted function f_{t-1} , and, by principal pivoting, finds out if f_{t-1} is convex, concave or neither.

<u>Convex (or concave) programming phase</u> (STEP 4) : Knowing that the programming subproblem

minimize $f_{t-1}(x)$, subject to $x \in F_{t-1}$

is either <u>convex</u> or <u>concave</u>, one finds the optimal solution(s), by applying the respective algorithm.

<u>Ordering and choice routines</u> : Each array I[t;] is constructed and arranged in an arbitrary but fixed way and the choice in STEP 6 amounts to a well-defined but arbitrary order in which the faces of P are generated and examined.

In fact, the arborescence constructed by the algorithm is guided by the following two rules:

<u>Forward choice rule</u> (STEP 3): Given a (n-t)-dimensional affine space Aff (F_{t-1}) for which the convexity test has indicated that f_{t-1} is neither <u>convex nor concave</u>, STEP 3 chooses one (the "first" one) (n-t-1)-dimensional affine space which is a subspace of Aff (F_{t-1}) ; clearly, because of the arbitrary order in I[t;] , this choice is well-determined but arbitrary.

<u>Backtrack-choice rule</u> (STEP 6): Given a (n-t)-dimensional affine space Aff (F_{t-1}) where the function f (i.e. f_{t-1})

is *either convex or concave*, we know that the current branch can be terminated because the optimal solution corresponding to that affine space is found in STEP 4; here STEP 6 chooses the next *collateral* (n-t)-dimensional affine space, increasing the pointer $m[t-1]$ by one; if there is no such set (i.e. $m[t-1] = m_I[t-1]$) , STEP 6 backtracks to the previous level, reducing t by one, and chooses the next collateral (n-t-1)-dimensional affine space. Here again, the sequence in which affine spaces are chosen is determined by the order within the index sets $I[t-1;]$ and $I[t-2;]$, and it is therefore well-defined but arbitrary.

The reader familiar with branch and bound procedures will recognize here the immediate possibility for eventual improvements of the overall efficiency by making use of a more elaborate choice rule which selects every element of the index sets $I[t,]$ *exactly once* but in a certain order of preference defined by additional computations. To be efficient this technique must be combined with the *bound estimates* mentioned in section 3 and no effort is made to go into the art involved by such an approach.

Footnotes

1/An interesting survey article [14] on *quasi-convex* and *pseudo-convex* by W. C. Mylander is to appear shortly in *Operations Research*; these are special cases of non-convex quadratic programming which can still be solved in very much the same way as convex quadratic programs by some algorithms. These cases may possess *one negative eigenvalue*, while the general case considered here may have arbitrary eigenvalues (> 0 , $= 0$, or < 0). In [14], Mylander reports an extension of Lemke's algorithm to the non-convex case.

2/In [12], Mueller presents a randomly generated heuristic approach, making use of properties similar to those presented in section 2.

3/A closely related approach has been suggested in [13] and is briefly compared with the present one at the end of section 3.

4/A similar result has been proved by Mueller in [12].

References

[1] Balas, E.: "The intersection cut - a new cutting plane for integer programming," Operations Research, Vol. 19, No.1, pp. 19-39 (1971).

[2] Balas, E. and C.-A. Burdet: "On concave quadratic programming," under preparation.

[3] Burdet, C.-A.: "Enumerative cuts I and II," submitted to Operations Research.

[4] Burdet, C.-A.: "Generating all the faces of a convex polyhedron," Carnegie-Mellon Management Science Report (1971).

[5] Cottle, R. W. and W. C. Mylander: "Ritter's cutting plane method for non-convex quadratic programming," in Integer and Non-linear Programming (J. Abadie, Ed.), North-Holland (1970).

[6] Glover, F.: "Cut-search methods in integer programming," University of Colorado (1970).

[7] Konno, H.: "Bilinear programming," Operations Research Department, Stanford University (1971).

[8] Raghavachari, M.: "On connections between zero-one integer programming and concave programming under linear constraints," Operations Research, Vol. 17, No. 4, pp. 680-684 (1969).

[9] Ritter, K.: "Stationary points of the quadratic maximum problem," Zeitschrift der Wahrscheinlichkeitstheorie und verwandte Gebiete, Bd 4 , pp. 149 - 158 (1965) .

[10] Ritter, K.: "A method for solving maximum problems with non-concave quadratic objective function," Zeitschrift der Wahrscheinlich--keitstheorie und verwandte Gebiete, Bd 4 , pp. 340 - 351 (1966)

[11] Tui, Hoang: "Concave programming under linear constraints," Soviet Mathematics, pp. 1437-1440 (1964).

[12] Mueller, R. K.: "A method for solving the indefinite quadratic programming problem," Management Science (Theory), Vol. 16, No. 15, pp. 333-339 (1970).

[13] Murty, K. G.: "On the finite nature of quadratic programming - A note on R. K. Mueller's paper," Department of Industrial Engineering, University of Michigan (1971).

[14] Mylander, W. C.: "Finite algorithms for solving quasi-convex quadratic programs," submitted to Operations Research (1971).