An Application of Lipschitzian Global Optimization to Product Design

E. M. T. HENDRIX[1] and J. PINTÉR[2]
[1]*Department of Mathematics, Agricultural University Wageningen, Dreijenlaan 4, 6703 HA Wageningen, The Netherlands*;
[2]*School for Recourse and Environmental Studies, Dalhousie University, 1312 Robie Street, Halifax, Nova Scotia, Canada.*

(Accepted 16 October 1991)

Abstract. The issue of finding feasible mixture designs is formulated and solved as a Lipschitzian global optimization problem. The solution algorithm is based on a simplicial partition strategy. Implementation aspects and extension possibilities are treated in some detail, providing also numerical examples.

Key words. Product development, best (or feasible) mixture, Lipschitzian optimization, simplicial partition strategies, implementation aspects.

1. Introduction

Branch-and-bound (B&B) methods have been proposed for solving a broad class of multiextremal (global) optimization problems, including Lipschitzian optimization. Description of the basic concepts and several implementation variants can be found, e.g., in Horst (1986), Horst and Tuy (1987, 1990) and Pintér (1986, 1988). We shall discuss here a Lipschitzian global optimization problem, originating from product design in mixing and processing industries. Specifically, in Section 2 product design problems are formulated that lead to finding feasible solutions of (generally speaking, indefinite) quadratic inequality systems. In Section 3 a B&B frame is presented to solve this problem. Section 4 compares different known and new lower bound estimates, applicable in the implementation of the B&B scheme. Section 5 is devoted to further implementation aspects, illustrated by numerical examples. Alternative solution approaches are briefed in Section 6, while Section 7 highlights extensions towards further applications of (general) Lipschitzian optimization on simplicial regions.

2. A Product Design Problem

The problem to be solved consists of identifying mixture products, each represented by a vector $x \in \mathbf{R}^n$, which meet certain requirements. The set of possible mixtures is mathematically defined by the unit simplex $S = \{x \in \mathbf{R}^n \mid \Sigma_j x_j = 1, x_j \geq 0\}$, in which the variables x_j express the rate of the components in product x.

Journal of Global Optimization **1**: 389–401, 1991.

Note that the set S lies in the $n-1$ dimensional hyperplane given by $\Sigma_j x_j = 1$.

The requirements are mathematically expressed by the inequalities $g_i(x) \leq 0$, $i = 1, \ldots, m$ derived from target quality levels, demands and properties. The functions g_i are assumed to be nonlinear smooth functions. This problem type is frequently posed with respect to products of pharmaceutical firms, chemical factories, oil refineries, etc. Specifically, the problem formulation and solution approaches presented here are applied at the product development department of a chemical company. We, in addition, assume that the functions g_i are quadratic. (Let us remark that these quadratic relationships can be derived on the basis of analytical or empirical second order approximations of smooth functions.)

The functions $g_i(x)$ can be written explicitly as

$$g_i(x) = x'Q_i x + d_i' x + c_i$$

in which Q_i is a symmetric n by n matrix, d_i is an n-vector and c_i is a scalar. In this way $D = \{x \in \mathbf{R}^n \mid g_i(x) \leq 0,\ i = 1, \ldots, m\}$ denotes the set which the "satisfactory" (feasible) products have to be elements of. Applying the above notation, the problem to be solved can be formulated as:

$$\text{Find an element of the set } D \cap S\,. \tag{1}$$

In practical situations, lower and upper bounds or other linear restrictions on x might also be present. Further on, process variables x_j that are not itself components of the mix, could be included in the problem formulation. Although these aspects can be implemented in the algorithm presented below, for simplicity, we avoid this here. Another important aspect is that the mixing industry is typically interested in stable solutions of (1), as irregularities, fluctuations may appear during the production process. Mathematically, this means that a subset of $S \cap D$ with a given volume ε should be looked for, or alternatively, an internal point of $S \cap D$ located at a given distance from the boundary of D is sought. This aspect will be briefly discussed later. The dimensions of the underlying practical problems are typically given by up to some 12 components and 10 properties.

Note that all g_i are Lipschitz continuous functions on S, therefore (1) can be solved by (see, e.g., in Horst and Thoai (1988)) minimizing over S the expression $\max_i \{g_i(x)\}$ or by minimizing $\Sigma_i [\max \{g_i(x), 0\}]^p$, $p > 0$. We present here an algorithm, which is closer in spirit to the infeasibility elimination idea suggested in Horst (1988) and in Pintér (1988).

As $g_i(x)$ are possibly indefinite quadratic functions, it is well-known from literature (see, e.g., Pardalos and Rosen (1987)) that the above minimization problems, equivalent to (1), are multiextremal Lipschitzian optimization problems. For illustration consider the following example.

EXAMPLE 1. A mixture product consisting of three components $n = 3$, has to be found which meets the following two requirements:

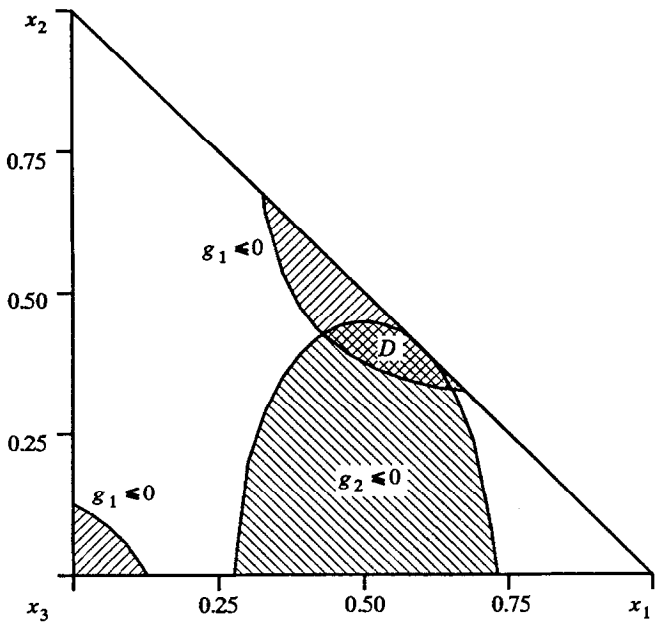

Fig. 1. The feasible set D of Example 1.

$$y_1(x) = -2 + 8x_1 + 8x_2 - 32x_1x_2 \qquad \leqslant -1$$

$$y_2(x) = \quad 4 - 12x_1 - 4x_3 + 4x_1x_3 + 10x_1^2 + 2x_3^2 \leqslant 0.4$$

Figure 1 forms a projection of S on the x_1, x_2 space. Vertex x_p represents a product x consisting for 100% of component x_p, $p = 1, 2, 3$. The area in which the feasible products are situated is given by D.

The problem of finding an element of $D \cap S$ is equivalent to minimizing the sum of the infeasibilities:

$$\min_{x \in S} g(x) := \min_{x \in S} \sum_i \max \{g_i(x), 0\} . \tag{2}$$

Problem (2) has a local optimum, e.g., in $x_{\text{loc}} = (0.125, 0, 0.875)$, $g(x_{\text{loc}}) = 0.725$, and of course a global optimum (=0) for all elements of $D \cap S$.

3. The Solution Algorithm

The algorithm suggested to solve problem (1) is based on a partition of S via adaptively generated subsets C_k. The sets C_k are simplices (having n vertices) of the $n-1$ dimensional space containing S. The information obtained in the n vertices of C_k is used to calculate lower bounds for the functions g_i. The lower bounds can be used to (possibly) eliminate subsets and to decide on which subset is to be splitted further. The algorithm is summarized by the following scheme.

ALGORITHM (A):

0. Set $C_1 := S$, $r := 1$

   At every iteration there exists a list of subsimplices of S: $C_1, \ldots, C_r$

   1. For every $i = 1, \ldots, m$ and $k = 1, \ldots r$ calculate lower bounds φ_{ik} of $g_i(x)$ on C_k: $\varphi_{ik} \leq g_i(x)$ $x \in C_k$
   2. If $\varphi_{ik} > 0$ for any i, delete C_k and set $r := r - 1$
      If the list of subsets is empty ($r = 0$), STOP: there is no feasible composition; in the opposite case, proceed to Step 3.
   3. Split subset C_k with the lowest value of $\Sigma_j \phi_{ik}$ into two parts of equal volume, over the longest edge; evaluate x_{new}, the midpoint of the longest edge of C_k. If $g_i(x_{\text{new}}) \leq 0$ for $i = 1, \ldots, m$, STOP: a solution is found. Otherwise delete the old C_k and add the two new subsets to the list, $r := r + 1$; return to 1.

Algorithm (A) can be seen as a special case of the branch-and-bound methods discussed, e.g., by Horst and Tuy (1990) and by Pintér (1988). If the fact that, in practice, one is interested in robust solutions, is implemented in (A) by deleting at Step 2 also subsets C_k which are too small ($\delta(C_k) < \varepsilon$), then it is clear that (A) converges to either a solution of (1), or produces the answer that there is no robust solution. Unfortunately, considering only this termination criterion, an exponential number of iterations (in n) may theoretically be necessary to verify that all subsets are smaller than ε.

Let us remark that algorithm (A) can be modified in such a way that it is not necessary to calculate all lower bounds at every iteration. The check in Step 2 can be interpreted as the question: is it possible that C_k contains a vector x for which $g_i(x) \leq 0$? To answer this question, it is not necessary to determine the lower bound φ_{ik}, if for one of the vertices x_{pk} of C_k holds $g_i(x_{pk}) \leq 0$. Algorithm (A) easily can be modified so that in such situations the lower bound is not calculated. Note that – as a consequence – the selection criterion in Step 3 should be changed accordingly.

4. Calculation of Lower Bounds

The lower bounds in Step 1 of an implemented algorithm can be derived, making use of the fact that g_i is a Lipschitz continuous function. Let x_{pk} denote the vertices of C_k, $p = 1, \ldots, n$. The value of φ_{ik} can be based on the following relations:

$$g_i(x) \geq g_i(x_{pk}) - L_{ik} \| x - x_{pk} \|, \quad x \in C_k, \quad p = 1, \ldots, n. \tag{3}$$

In (3) L_{ik} is the Lipschitz-constant of g_i on C_k. The Lipschitz-constant L_{ik} can be (over)estimated by solving

$$L_{ik} = \max_{x \in C_k} \| \nabla g_i(x) \|. \tag{4}$$

As the functions g_i are quadratic, problem (4) means the maximization of a convex function over a polyhedron; hence it can be solved by simply evaluating $\| \nabla g_i(x) \|$ at every vertex x_{pk}. Note that the estimate of L can be made sharper, by projecting first the gradient $\nabla g_i(x)$ on the hyperplane of S. In the practical implementation of the algorithm we applied the lower bound φ_{ik}, based on (3) and (4):

$$\phi_{ik} = \max_{x_{pk}} \{ g_i(x_{pk}) - L_{ik} \max_{x_{vk}} \| x_{vk} - x_{pk} \| \}$$
(x_{vk} being the vertices of C_k, different from x_{pk}). (5)

In the present context, let us discuss concisely some alternative ways to determine lower bounds, based either on the (sole) Lipschitzian or on the quadratic structure of the problem.

In Lipschitzian global optimization on interval (box) regions, Pintér (1986, 1988) forms rectangular subsets $[a_k, b_k]$, for which the information available is based on the "lower-left" vertex a_k and the "upper-right" vertex b_k. His approach can be termed a diagonal extension of known univariate methods, e.g., of the Danilin–Piyavskii–Shubert method (cf. the references). The selection of the subset to be refined (cf. Step 3) is based on the (rectangular subset) selector function (Pintér, 1986):

$$(g_i(a_k) + g_i(b_k))/2 - L_i \| a_k - b_k \| .$$

For a lower bound used in Step 2 of the algorithm (elimination), Pintér (1988) uses the expression

$$\max \{ g_i(a_k), g_i(b_k) \} - L_i \| a_k - b_k \| .$$

Applying directly this idea to the simplicial algorithm given above would lead to the lower bound:

$$\phi_{ik} = \max_{x_{pk}} \{ g_i(x_{pk}) \} - L_{ik} \delta(C_k) , \tag{6}$$

where $\delta(C_k) = \max \{ \| x_{vk} - x_{pk} \| : v \neq p \}$ is the diameter of C_k.

Observe that this would yield a more crude estimate that the lower bound given by (5); on the other hand, (6) requires somewhat less calculation per iteration than (5). For practical implementations several interior points of C_k can be additionally evaluated and included into the estimations (5) or (6), usually improving the bounds.

The sharpest lower bound given all information of (3) can be found by solving explicitly the following problem:

$$\begin{aligned} &\underset{x,z}{\text{minimize}} \{ z \} \\ &\text{subject to} \\ &x \in C_k \\ &z \geqslant g_i(x_{pk}) - L_{ik} \| x - x_{pk} \| \quad p = 1, \ldots, n . \end{aligned} \tag{7}$$

Meewella and Mayne (1988) approximate (7) by a piecewise linear problem, replacing $\| \cdot \|$ by the infinite norm. They apply further rectangular subsets C_k storing all 2^n-vertices. In every iteration they solve 2^n-*LP* problems.

The n-dimensional variant of (7), in which C_k is defined by $n+1$ vertices, defines the problem of finding the deepest point of a "pommes-frites bag" and was studied a.o. by Mladineo (1986). The minimizer x of (7) in most cases can be found by solving n linear equations given by:

$$(x_p - x_t)'x = (x_p - x_t)'\left[\frac{x_p + x_t}{2} + \frac{f_p - f_t}{2L_{ik} \| x_p - x_t \|}(x_t - x_p)\right], \quad p \neq t,$$

in which $x_t = \text{argmin}_{x_p} \{g_i(x_p)\}$, x_p the vertices of simplex C_k, $p = 1, \ldots, n+1$. Solving (7) is less laborious, if regular simplices are used. Relations for this can be found in the bracketing procedures and geometrical observations of Wood (1991) and Baritompa (1991).

In calculating the lower bound, one could make use of the fact that $g_i(x)$ is a quadratic function. If Q_i is positive semi-definite, then the minimum of $g_i(x)$ over C_k can be found by a convex programming algorithm. In the situation when Q_i is negative semi-definite, $g_i(x)$ is concave, a linear underestimation can be given, see Horst (1986), Pardalos and Rosen (1987). The fact that Q_i might be indefinite makes the problem of finding a lower bound more complex. A possible approach is to "make" $g_i(x)$ convex by replacing $x'Q_ix$ by $l_i \| x \|^2$, in which l_i is the largest eigenvalue of Q_i. A convex minorant (subfunctional) Θ_{ik} can be found in the following way. Let X_k be the n by n matrix with the vertices of C_k as columns. The minorant:

$$\Theta_{ik}(x) = \beta'_{ik}x + l_i \| x \|^2$$

can be found by determining $\beta_{ik} = (X_k^{-1})'h_k$, in which the n-vector h_k gives the difference between $g_i(x)$ and the convex function $l_i \| x \|^2$, with components

$$h_{pk} = g_i(x_{pk}) - l_i \| x_{pk} \|^2 .$$

The minorant $\Theta_{ik}(x)$ equals to $g_i(x)$ in the vertices of C_k, further on, it is convex. The lower bound can then be calculated by determining the minimum of Θ_{ik} over C_k. It is clear that this way of determining the lower bound requires the solution of convex programming problems at every iteration.

A more sophisticated elaboration of this idea is due to Pardalos *et al.* (1987). The quadratic indefinite objective $x'Qx$ is seen as a separable function by considering $Q = UDU'$, in which D is the diagonal matrix of eigenvalues and U consists of the n orthonormal eigenvectors. The concave part can be underestimated by an affine minorant and the convex part is left unchanged. In the algorithm of Pardalos *et al.*, at every iteration a convex envelope constructed in this way is minimized by the MINOS optimization system. Observe that for the m quadratic functions defining problem (1), this approach might require many local optimization steps.

Another piecewise linear underestimation of (possibly) indefinite quadratic functions can be found in the overview of Al-Khayyal (1990) on bilinear problems. One can make use of the fact that the objective of the following problem:

$$\begin{aligned}&\min \{x_1 x_2\}\\&\text{subject to}\\&l_1 \leq x_1 \leq L_1\\&l_2 \leq x_2 \leq L_2\end{aligned}$$

can be minorated by:

$$\begin{aligned}&\min \{z\}\\&\text{subject to}\\&l_j \leq x_j \leq L_j \quad j = 1, 2\\&z \geq l_1 x_2 + l_2 x_1 - l_1 l_2\\&z \geq L_1 x_2 + L_2 x_1 - L_1 L_2 .\end{aligned}$$

This idea can be used in general biconvex problems as shown by Al-Khayyal and Falk (1983). Again, application of this idea in algorithm (A) would require the solution of linear programming problems in every iteration. Foulds, Haugland and Jörnsten (1991) show the applicability of this sequential *LP* concept for the bilinear forms in the pooling problem in petrochemical industry. The bilinear term in the problem described in their paper originates from the balance equations of the concentrations in the pooled product and not in the interaction between mixed products as described in Section 2.

5. Implementation Aspects

The performance of algorithm (A) is illustrated and some implementations aspects are discussed below, via further examples.

EXAMPLE 2. Figure 2 gives the partition of S for the problem introduced in Example 1. After 29 iterations, 26 points have been evaluated, a feasible point in D is found and two subsets have been deleted. The feasible point found is $x^* = (0.5,\ 0.375,\ 0.125)$ with "acceptable" properties expressed by $y_1(x^*) = -1$ and $y_2(x^*) = 0.281$.

One of the problems of implementing a branch-and-bound algorithm in a computer program is that information concerning the partition sets has to be kept in the computer memory; this can be done by maintaining a list of subsets and a list of points generated, but special data structures based, e.g., on vertices or edges are also possible. The problem is that the program should not run out of memory, before a solution has been found. In branch-and-bound methods effi-

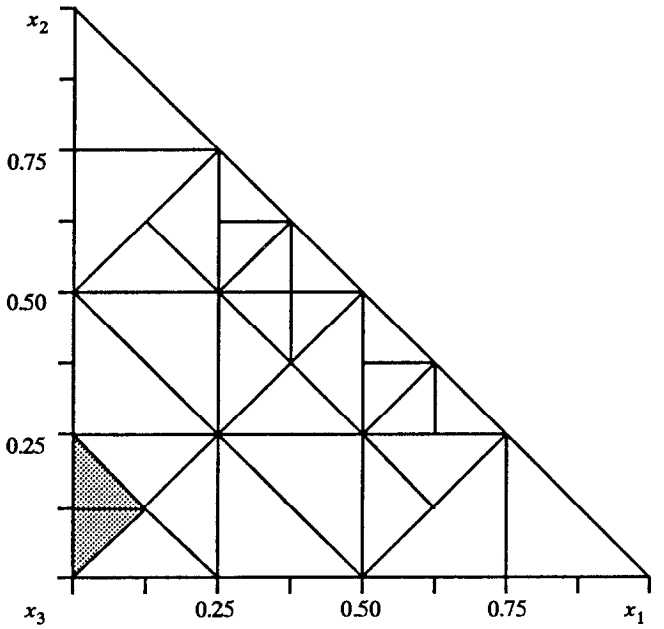

Fig. 2. Partition after 29 iterations.

cient use can be made of the fact that memory can be "recycled" if partition sets are deleted from further consideration; this can be done, e.g., by applying linked list structures.

Another aspect is that, by the symmetry of (A), a generated point may be used as a splitting point several times, as can be seen in Figure 2. It is important that such a point is evaluated only once and will not be added again to the list of points, which occupies most of the memory. Applying this simple idea, only a single new point which is not already part of the search information, is to be evaluated at every iteration cycle.

According to our numerical experience, the algorithm suggested can solve problems with a few variables (say, up to $n = 5$) in several hundred iteration steps. This will be illustrated by the following example.

EXAMPLE 3. We consider a test problem originating from a practical application with $n = 3$ components and $m = 5$ properties. The coefficients of $y_i(x)$ can be found in the Appendix. Assume that the following requirements are given for the properties:

$$y_1 \leqslant 1.496\,, \quad y_2 \geqslant 0.92\,, \quad y_3 \leqslant 10\,, \quad y_4 \geqslant 179\,, \quad y_5 \leqslant 85\,.$$

Applying (A) a feasible solution is found after 80 new points have been

generated. The solution found is given by:

$$x = (37.5, 43.75, 18.75) \text{ and } y(x) = (1.48,\ 0.921,\ 9.114,\ 180.77,\ -42.81)$$

Naturally, it is possible to generate more feasible points by not terminating the algorithm. For the problem given, this action required 100 points to be evaluated additionally, in order to find 10 more feasible points. Note that it can be more easy to carry out an "exhaustive" search say, in a ball around the first solution found to estimate how large one of the connected components of the set $S \cap D$ is.

The fact that the number of iterations becomes larger, if the minimum of $g(x)$ (cf. (2)) is a small positive number, can be illustrated by modifying the above example as follows:

Let $0 \leq y_1 \leq 85$, $y_2 \geq 0.963$; the other requirements are left unchanged. The algorithm needs 56 iterations to conclude that the problem does not have a solution. If the problem is "near to feasibility", e.g., $y_2 \geq 0.94$, then concluding that there is no solution needs 157 iterations; if $y_2 \geq 0.93$, the "no solution" conclusion requires 239 iterations. having these simple examples in mind, it may be intuitively clear that such "bad" cases can be very hard to solve in higher dimensions.

6. Some Alternative Approaches to Solve (1)

The discussed variants of the branch-and-bound algorithm make use of the quadratic and/or Lipschitzian structure of (1). The problem might also be tackled by other exhaustive search techniques; some of them will be highlighted below.

Note first that one should be aware of the fact that an indefinite quadratic programming problem in the worst case possesses 2^p, local minima, where p is the number of negative eigenvalues of Q (see Pardalos and Rosen (1987)); moreover, (1) is defined by m quadratic functions. Note further than an objective function equivalent to (1) such as $g(x)$ (cf. (2)) is not differentiable, so that local search methods based on gradients should take this into account.

Manas (1968) proposes a kind of grid search method for solving the indefinite quadratic programming problem. For problem (1), a minimum size for the grid can be identified by the Lipschitz-constant $g_i(x)$ on S and by the minimum acceptable size of a subinterval ε. Naturally, the number of grid points grows exponentially with n. Another viable alternative is to make use of random search techniques and extensions for solving the problem, see, e.g., Rinnooy Kan and Timmer (1987a, b). However, if after any finite number of iterations no solution is found, then it is not necessarily clear whether (1) has got a solution or not. Note at the same time that grid search as well as random search methods generate new points that can be evaluated much quicker than algorithm (A). This is caused by the fact that (A) calculates gradients and longest edges in every iteration and

that newly generated candidate solutions may turn out to be already evaluated. Therefore direct search methods may be faster in finding a solution, if the set $S \cap D$ is relatively large (while might perform poorly in the opposite case). In many practical situations, during the actual decision making (mixture design) procedure the requirements towards the products are sequentially increased, thus making the feasible region gradually smaller. This comment seems to validate our approach (or similar ones). Let us remark additionally that in the practically relevant case of infeasibility, the approach suggested aims at a "best compromise" infeasible solution, in the sense of the objective form $g(x)$ suggested (cf., e.g., (2)).

7. Further Application Prospectives

Consider the global optimization problem

$$\min_{x \in S} f(x) \tag{8}$$

in which f is Lipschitzian on the simplex S, $L > 0$ being a valid (over) estimate of its Lipschitz-constant. We shall assume furthermore that f has at most a countable set X^* of (isolated) global minimizers on S.

In a number of practical cases, the "exact" analytical dependence of f on x is not known: instead of that, the "quality" of x is to be evaluated experimentally or algorithmically. This way, although the smooth analytical behaviour of f may be known, the use of further structural and/or higher information may be out of question or its use can be very tedious. In such cases, Lipschitzian optimization can be rationally applied for solving (8).

To highlight two problem-classes that can be described in the form (8), one can mention, e.g., "optimal" mixture design and "optimal" negotiated combinations of expert opinions: for details, see, e.g., Klafszky, Mayer and Terlaky (1989) and Pintér and Cooke (1987), respectively. The work of Klafszky *et al.* considers the problem of minimizing the "discrepancy" between mixture x and a prespecified "ideal' target product. In particular, they show that if both the base materials and the target can be modelled by discrete probability distributions and the function f is defined by any of several statistical divergence functions (Csiszár (1975)), then the problem derived is solvable by corresponding nonlinear programming techniques. A straightforward extension of their approach leads to the general model (8).

In Pintér and Cooke (1987) an optimization approach is presented for aggregating individual expert "opinions" – modelled as point values, real vectors or probability distributions – in some quantitatively "best" sense: the framework proposed there subsumes a large variety of context-dependent realizations. Again, this general problem-type can frequently be modelled in the form (8).

As it is known – following, e.g., the conceptual framework of Horst and Tuy

(1987, 1990) or Pintér (1987, 1988) – convergent B&B strategies can be applied to solve (8). These methods, tailored to the problem-type (8), produce corresponding adaptive simplicial partitions in such a manner that the accumulation point set of the generated vertex set is identical to X^*. Different ways of realizing convergent B&B schemes are indicated in Horst and Tuy (1990, chapters IV and XI) and in Pintér (1990). Specifically, in the latter paper subsimplex-specific Lipschitzian bounds, similar to (5), are also derived.

Let us remark finally that the solution approaches mentioned can be directly extended to the case, in which a finite number of Lipschitzian constraints $g_i(x) \leq 0$, $i = 1, \ldots, m$ (that determine a non-empty, robust feasible region) is added to the explicit constraint set $x \in S$ in (8): for details, cf. the works referred to above.

8. Conclusions

In this paper an algorithm was proposed to find feasible solutions of indefinite quadratic constrained problems on an imbedding simplex; as it is shown, problem (1) can be handled in the frames of global optimization. The suggested algorithm is based on the branch-and-bound concept and theoretically converges to solutions of (1) or concludes that there is no feasible solution. The algorithm is simple and makes less use of the fact that the constraints are quadratic than earlier algorithms of Pardalos *et al.* and of Al-Khayyal for indefinite quadratic programming; furthermore, it can be extended for solving (more general) global optimization problems on simplicial feasible regions. For the implementation of branch-and-bound algorithms, linked list structures or other dynamic memory allocation can be suggested. The numerical examples presented indicate the viability of the approach.

Acknowledgement

Special thanks are due to Ms. H. de Blank (Agricultural University Wageningen) for programming the algorithm and accomplishing the test runs.

Appendix

The calculation of the property functions y_i in Example 3 is based on the following relations. Property y_i is calculated as:

$$\begin{aligned} y_i(x) = c_i &+ d_{i1}\hat{x}_1 + d_{i2}\hat{x}_2 + d_{i3}\hat{x}_3 \\ &+ q_{i12}\hat{x}_1\hat{x}_2 + q_{i13}\hat{x}_1\hat{x}_3 + q_{i23}\hat{x}_2\hat{x}_3 \\ &+ q_{i11}\hat{x}_1^2 + q_{i22}\hat{x}_2^2 + q_{i33}\hat{x}_3^2 \end{aligned}$$

in which $\hat{x}_i = x_i - 0.5$.

The data that are used are given by the following table:

property i	c_i	d_{i1}	d_{i2}	d_{i3}	q_{i12}	q_{i13}	q_{i23}
1	1.495	−0.006	−0.024	0.050	−0.002	0.017	−0.021
2	0.951	−0.014	−0.048	0.108	−0.001	−0.004	0.006
3	15.986	4.729	−16.657	−8.974	−10.174	−21.977	−86.952
4	178.708	−0.687	12.800	−7.347	0.241	−4.947	−3.967
5	52.002	−5.217	−201.326	192.989	−6.180	337.106	1030.228

i	q_{i11}	q_{i22}	q_{i33}
1	0.001	0.008	−0.021
2	0.004	0.001	−0.014
3	20.605	32.003	−81.278
4	−0.766	−0.528	7.822
5	116.750	−67.424	−845.215

References

Al-Khayyal, F. A. and Falk, J. E. (1983), Jointly Constrained Biconvex Programming, *Mathematics of Operations Research* **8**, 273–286.

Al-Khayyal, F. A. (1990), Jointly Constrained Bilinear Programs and Related Problems: An Overview, *Computers & Mathematics with Applications* **11**, 53–62.

Baritompa, W. (1991), Customizing Methods for Global Optimization: A Bisection Viewpoint. Preprint, University of Canterbury, New Zealand.

Csiszár, I. (1975). I-Divergence Geometry of Probability Distributions and Minimization Problems, *Annals of Probability* **3**, 146–158.

Danilin, Yu. and Piyavskii,, S. A. (1967), An Algorithm for Finding the Absolute Minimum, in: *Theory of Optimal Decisions* **2**, 25–37, Institute of Cybernetics of the Ukrainian Academy of Sciences. (In Russian.)

Foulds, L. R., Haugland, D., and Jörnsten, K. (1991), A Bilinear Approach to the Pooling Problem. Working paper no 90/03 Chr. Michelsen Institute, Centre for Petroleumeconomics, Bergen, Norway. To appear in *Mathematical Programming*.

Horst, R. (1986), *A General Class of Branch-and-Bound Methods in Global Optimization with Some New Approaches for Concave Minimization, J. of Optimization Theory and Applications* **51**, 271–291.

Horst, R. and Tuy, H. (1987), On the Convergence of Global Methods in Multiextremal Optimization, *J. of Optimization Theory and Applications* **54**, 253–271.

Horst, R. and Thoai, Ng. V. (1988), Branch-and-Bound Methods for Solving Systems of Lipschitzian Equations and Inequalities, *J. of Optimization Theory and Applications* **58**, 139–145.

Horst, R. (1988), Deterministic Global Optimization with Partition Sets whose Feasibility Is Not Known: Application to Concave Minimization, Reverse Convex Constraints, DC-Programming and Lipschitzian Optimization, *J. of Optimization Theory and Applications* **58**, 11–37.

Horst, R. and Tuy, H. (1990), *Global Optimization (Deterministic Approaches)*, Springer, Berlin.

Klafszky, E., Mayer, J., and Terlaky, T. (1989), Linearly Constrained Estimation by Mathematical Programming, *European Journal of Operational Research* **34**, 254–267.

Manas, M. (1968), An Algorithm for a Nonconvex Programming Problem, *Economic. Mathem. Obzor* **4**, 202–212.

Meewella, C. C. and Mayne D. Q. (1988), An Algorithm for Global Optimization of Lipschitz Continuous Functions, *J. of Optimization Theory and Applications* **57**, 307–322.

Mladineo, R. H. (1986), An Algorithm for Finding the Global Maximum of a Multimodal, Multivariate Functions, *Mathematical Programming* **34**, 188–200.

Pardalos, P. M., Glick, J. H., and Rosen, J. B. (1987), Global Minimization of Indefinite Quadratic Problems, *Computing* **39**, 281–291.

Pardalos, P. M. and Rosen, J. B. (1987), *Constrained Global Optimization: Algorithms and Applications*, Springer, Berlin.

Pintér, J. (1986), Extended Univariate Algorithms for n-Dimensional Global Optimization, *Computing* **36**, 91–103.

Pintér, J. (1986), Extended Univariate Algorithms for n-Dimensional Global Optimization, Research Report **87–61**, Department of Mathematics and Informatics, Delft University of Technology (revised version to appear in *Mathematical Programming*).

Pintér, J. and Cooke, R. (1987), Combining Expert Opinions: An Optimization Framework, Research Report **87–84**, Department of Mathematics and Informatics, Delft University of Technology.

Pintér, J. (1988), Branch-and-Bound Algorithms for Solving Global Optimization Problems with Lipschitzian Structure, *Optimization* **19**, 101–110.

Pintér, J (1990), Simplicial Partition Strategies, for Lipschitzian Global Optimization. Preprint, Institute for Inland Water Management and Waste Water Treatment, Lelystad, The Netherlands.

Rinnooy Kan, A. H. G. and Timmer, G. T. (1987), Stochastic Global Optimization Methods: I. Clustering Methods, *Mathematical Programming* **39**, 27–56.

Rinnooy Kan, A. G. H. and Timmer, G. T. (1987), Stochastic Global Optimization Methods: II. Multilevel Methods, *Mathematical Programming* **39**, 57–78.

Shubert, B. O. (1972), A Sequential Method Seeking the Global Maximum of a Function, *SIAM Journal of Numerical Analysis* **9**, 379–388.

Wood, G. R. (1991), The Bisection Method in Higher Dimensions, Preprint, University of Canterbury, New Zealand. To appear in *Mathematical Programming*.