J Glob Optim (2010) 47:211–231
DOI 10.1007/s10898-009-9467-2

Bounded lower subdifferentiability optimization techniques: applications

Gleb Beliakov · Albert Ferrer

Received: 20 January 2007 / Accepted: 12 August 2009 / Published online: 17 September 2009

Abstract In this article we develop a global optimization algorithm for quasiconvex programming where the objective function is a Lipschitz function which may have "flat parts". We adapt the Extended Cutting Angle method to quasiconvex functions, which reduces significantly the number of iterations and objective function evaluations, and consequently the total computing time. Applications of such an algorithm to mathematical programming problems in which the objective function is derived from economic systems and location problems are described. Computational results are presented.

Keywords Lipschitz and quasiconvex programming · Global optimization · Cutting angle method · Abstract convexity

Mathematics Subject Classification (2000) 90C26 · 90C30

1 Introduction

Quasiconvex programming, i.e., programs with a quasiconvex objective function and convex constraints, has been widely applied to modeling in management, decision making, information theory, applied mathematics and physics. In particular, some location problems [8,9], and modeling economic systems via fractional programming [21,15], arise as specific instances of quasiconvex programming.

The research of Albert Ferrer has been partially supported by the Ministerio de Ciencia y Tecnología, Project MCYT, DPI 2005-09117-C02-01.

G. Beliakov (✉)
School of Information Technology, Deakin University, Melbourne, VIC, Australia
e-mail: gleb@deakin.edu.au; gbeliako@gmail.com

A. Ferrer
Departament de Matemàtica Aplicada I, Universitat Politècnica de Catalunya, Barcelona, Spain
e-mail: alberto.ferrer@upc.edu

If a quasiconvex function is differentiable and the gradient is not zero, except the minimizer, it is possible to solve quasiconvex programs by using standard algorithms. On the other hand, for nonsmooth quasiconvex programs a well-developed theory exists, which is based on the concept of lower subgradient of a boundedly lower subdiferentiable function, which is a quasiconvex and Lipschitz function (see [10,18]).

Definition 1.1 A real-valued function f defined on $X \subset I\!R^n$ is called lower subdiferentiable at $z \in X$ if there exists $p \in I\!R^n$, which is called a lower subgradient of f at z, such that

$$f(x) \geq f(z) + \langle p,\ x - z\rangle,\ \text{for every } x \in f^{-1}(-\infty, f(z)), \tag{1}$$

The set of all the lower subgradients at z is denoted by $\partial^- f(z)$. A function is lower subdifferentiable (*lsd*) on X if it admits at least one lower subgradient at each point of X. Moreover, a function f is called boundedly lower subdiferentiable (*blsd*) on X, if, there exists a positive number $r \in R$, named a *blsd*-bound of f, such that at each point $z \in X$, $\partial^- f(z) \cap B(0, r) \neq \emptyset$, where $B(0,\ r) := \{x \in I\!R : ||x|| \leq r\}$.

Theorem 1.1 (see [18], p. 41) *Let f be a bounded above real-valued function on $X \subset I\!R^n$. Then, f is blsd on X if and only if there exists a quasiconvex Lipschitz function on $I\!R^n$ extending f.*

In this theory, the formulation of quasiconvex optimization appears as a natural generalization of the Cutting Plane method by Kelley [17] and Cheney and Golstein [11] to solve convex programs, which is based on building lower approximations (underestimates) to the objective function f by a pointwise maximum of elements of the subgradient of f (support hyperplanes). In convex programming, the underestimate is a piecewise linear convex function and its minimization can be performed by using linear programming techniques.

In contrast to the convex programs, application of the generalized Cutting Plane method to quasiconvex programs is problematic, because even knowing a subgradient of the objective function at a point, it is not generally possible to obtain a lower subgradient at this point. Moreover, having solved the problem of finding a lower subgradient we then come up against another even more complicated problem if the objective function has "flat parts". Even though quasiconvex functions have a single local (and hence global) minimum, their numerical optimization is as challenging as that of multiextremal functions. Take for example a quasiconvex function $f(x) := \min\{1, M||x - a||\}$ on $X \subset I\!R^n$ for some $a \in I\!R^n$, whose Lipschitz constant is M, and assume $\text{diam}(X) \gg \frac{1}{M}$. To locate its minimum, a systematic exploration of the domain X is required, because evaluating f and its gradient at any x (unless x is close to a) provides no useful information about location of the minimum; such exploration is impractical even for moderate dimension.

In this article we present a contribution to developing a global optimization algorithm for quasiconvex programming problems, whose objective function is a Lipschitz function that may have "flat parts". We build a global optimization algorithm for quasiconvex functions based on the theory of abstract convexity [20], whose elements are outlined in Sect. 2. In Sect. 3 we start with the Extended Cutting Angle method (ECAM), which is based on the theory of abstract convexity. This method is applicable to global optimization of any Lipschitz function, but it does not take into account quasiconvexity. In Sect. 4 we present a new optimization method for quasiconvex programs with a boundedly lower subdifferentiable objective function, whose graph may have flat parts. Our main contribution is to adapt ECAM for quasiconvex functions, and we call the adapted method QECAM. This adaptation allows us to improve the efficiency of the method by deleting parts of the feasible domain in which the global minimizers cannot reside and we give a method to trascend the "flat parts" in the graph

of the objective function. In Sect. 5 we present applications of this algorithm to mathematical programming problems derived from economic systems and location problems. Finally in Sect. 6 we discuss computational results and state the conclusions.

2 Abstract convexity and quasiconvex programs

2.1 Minkowski gauge and a distance function

Let P be a bounded star-shaped set in $\mathbb{R}^n$ that includes the origin in its interior. If $x \in P$ and $\lambda \in [0, 1]$ then $\lambda x \in P$ [20]. Its Minkowski gauge is the function

$$\mu_P(x) := \inf\{\lambda > 0 \mid x \in \lambda P\}.$$

It enjoys several interesting properties, as reported in [12,20], in particular it is defined on $\mathbb{R}^n$, is non-negative and positively homogeneous of degree one. If P is convex, then $\mu_P(x)$ is also convex and sublinear. That is, for a convex P we have [12,20]

(1) $\mu_P(\lambda x) = \lambda \mu_P(x), \forall x \in \mathbb{R}^n, \forall \lambda > 0$;
(2) $\mu_P(x + y) \leq \mu_P(x) + \mu_P(y), \forall x, y \in \mathbb{R}^n$;
(3) $\exists c_1, c_2$ such that $c_1||x|| \leq \mu_P(x) \leq c_2||x||, \forall x \in \mathbb{R}^n$.

The function $d_P(x, y) := \mu_P(x - y)$, defined with the help of Minkowski gauge, verifies $d_P(x, x) = 0$ for all x and the triangular inequality but d_P needs not be symmetric, hence it is not necessarily a metric (it is called a quasi-metric). In this article for simplicity we refer to d_P as a distance.

In the following we will need the simplicial distance defined by the simplex centered at 0, which is the intersection of $n + 1$ halfspaces

$$P := \bigcap_{i=1}^{n+1} \{x : \langle x,\ h_i \rangle \leq 1\}, \tag{2}$$

with $\langle \cdot, \cdot \rangle$ denoting the inner product and $h_i \in \mathbb{R}^n$ are the directional vectors

$$\begin{aligned} &h_1 := (-1, 0, 0, \ldots),\\ &h_2 := (0, -1, 0, \ldots),\\ &\ldots\\ &h_n := (0, \ldots, 0, -1),\\ &h_{n+1} := (1, 1, \ldots, 1). \end{aligned} \tag{3}$$

Then

$$\mu_P(x) = \max\{\langle x,\ h_i \rangle : 1 \leq i \leq n + 1\},$$

and

$$d_P(x, y) = \max\{\langle x - y,\ h_i \rangle : 1 \leq i \leq n + 1\},$$

which can be written in the form

$$d_P(x, y) = \max\left\{ \max_{i=1,\ldots n} (y_i - x_i), \sum_{i=1}^{n} (x_i - y_i) \right\}. \tag{4}$$

Let us now for the purposes of convenience introduce a slack variable $x_{n+1}, x_{n+1} := 1 - \sum_{i=1}^{n} x_i$. In other words, we will consider vectors in the plane $\sum_{i=1}^{n+1} x_i = 1$ in $\mathbb{R}^{n+1}$. Of course, this plane is our original space $\mathbb{R}^n$. With the help of the new coordinate, and using

$$\sum_{i=1}^{n}(x_i - y_i) = 1 - \sum_{i=1}^{n} y_i - \left(1 - \sum_{i=1}^{n} x_i\right) = y_{n+1} - x_{n+1},$$

we can write (4) in a more symmetric form

$$d_P(x, y) = \max_{i=1,\ldots,n+1}(y_i - x_i). \tag{5}$$

2.2 Abstract convex functions

We present a few relevant results from [20].

Definition 2.1 A function f is abstract convex with respect to the set of functions H (or H-convex) if there exists $U \subset H$ such that

$$f(x) = \sup\{h(x) : h \in U\},\ \forall x \in X.$$

Definition 2.2 The set U of H-minorants of f is called the support set of f with respect to the set of functions H and denoted by $\mathrm{supp}(f, H)$. Then,

$$\mathrm{supp}(f, H) = \{h \in H, h \leq f\}.$$

Definition 2.3 An H-subgradient of f at x is a function

$$h \in H \text{ such that } f(y) \geq h(y) - (h(x) - f(x)),\ \forall y \in X.$$

The set of all H-subgradients of f at x is called H-subdifferential and denoted by $\partial_H f(x)$. Then,

$$\partial_H f(x) = \{h \in H \mid \forall y \in X, f(y) \geq h(y) - (h(x) - f(x))\}.$$

Definition 2.4 The set $\partial_H^* f(x)$ at x is defined as

$$\partial_H^* f(x) := \{h \in \mathrm{supp}(f, H) \mid h(x) = f(x)\}.$$

Proposition 2.1 [20], p. 10. *If the set H is closed under vertical shifts, i.e., $h \in H$ and $c \in \mathbb{R}$ implies $h - c \in H$, then $\partial_H^* f(x) = \partial_H f(x)$.*

Proposition 2.2 [20], p. 239. *Let X be a metric space with the metric ρ and H be the set of functions of the form*

$$h(x) := b - a\rho(x, z),$$

with $x, z \in X$, $a \geq 0$ and $b \in \mathbb{R}$. Then each lower semicontinuous function $f : X \to \mathbb{R}_{+\infty}$, such that $\inf_{x \in X} f(x) > -\infty$, is abstract convex with respect to the set H.

If ρ is a distance function d_P, we obtain that each lower semicontinuous function bounded from below is abstract convex with respect to the set of functions H of the form

$$h(x) := b - Cd_P(x, z),$$

with $x, z \in X = I\!R^n$, $C \geq 0$ and $b \in I\!R$, where d_P is a distance function defined by $d_P(x, y) = \mu_P(x - y)$. Further, the H-subdifferential is not empty for each Lipschitz function f and $\partial^*_H f(x) = \partial_H f(x)$ [5,6].

The implication of this result is that for each $z \in X$ and $C \geq M$, M the Lipschitz constant of f, the function $h(x) := f(z) - Cd_P(x, z)$ verifies that $h \in \partial^*_H f(x)$.

Let us relate quasiconvexity to the theory of abstract convexity. In the case of a *blsd* function f, let H be the set of functions

$$h(x) := \min\{a + \langle p,\ x - b\rangle, a\}, \tag{6}$$

with $a \in I\!R$ and $b \in R^n$. Then,

$$f(x) = \sup\{h(x) : h \in U\}, \tag{7}$$

where U is the subset of functions of H defined in the form:

$$h(x) := \min\{f(z) + \langle p,\ x - z\rangle, f(z)\}, \tag{8}$$

with $z \in X$ and $p \in \partial^- f(z) \cap B(0, r)$, i.e., f is H-convex in the sense of the abstract convexity.

3 Extended cutting angle method for Lipschitz functions (ECAM)

We start this section by recalling the Cutting Plane method by Kelley [17] and Cheney and Golstein [11] to solve convex programs. It is known that a lower semicontinous convex function f is the upper envelope of the set of all its affine minorants, compare to Definition 2.1:

$$f(x) = \sup\{h(x) \mid h \text{ affine function}, h \leq f\}. \tag{9}$$

The Cutting Plane method proceeds by constructing a piecewise affine underestimate of the objective function f as a pointwise maximum of elements of the subdifferential of f. At each iteration a new support hyperplane is built at the point of the global minimum of the current underestimate, which is found by methods of linear programming. Then the underestimate is modified by adding this new support hyperplane so it becomes a more accurate approximation to f, and the method iterates.

The generalized cutting plane method (of which ECAM is a special case) [20] follows the same script, except that the underestimate is built using H-subgradients. Consequently, minimization of an underestimate is no longer a convex problem. For special cases of abstract convex functions, specialized efficient numerical methods for minimizing the underestimates exist [1,2,4,20].

As we mentioned, *blsd* functions are quasiconvex Lipschitz functions. Let f be quasiconvex and Lipschitz, with its Lipschitz constant with respect to the distance d being M. The following algorithm specifies the steps of the generalized cutting plane method [20] to solve the programming problem

$$\min\{f(x) \mid x \in S\}, \tag{10}$$

with S the unit simplex.

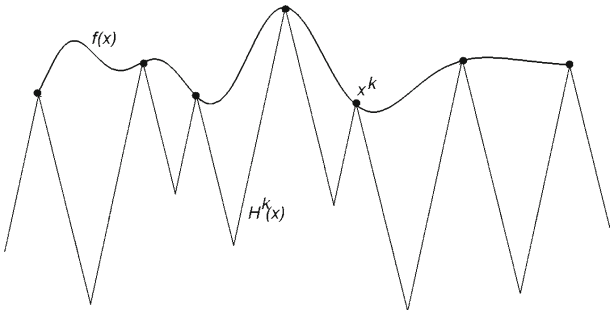

Fig. 1 The saw-tooth underestimate $H^K(x)$ in the univariate case

Algorithm 3.1 ***Generalized cutting plane algorithm***
K_{max} denotes the maximum number of iterations allowed, f_{best} is the best objective value obtained until the last performed iteration and ϵ is the required precision.

Step 0. (Initialisation)

0.1 Set $K = 1$.
0.2 Choose an arbitrary initial point $x^1 \in S$.

Step 1. (Calculate H-subdifferential)

1.1 Calculate $h^K \in \partial^*_H f(x^K)$.
1.2 Define $H^K(x) := \max_{k=1,\ldots,K} h^k(x)$, *for all* $x \in S$.

Step 2. (Minimize H^K*)*

2.1 Solve Problem minimize $H^K(x)$, *s.t.* $x \in S$. *Let* x^* *be its solution.*
2.2 Set $K := K + 1$, $x^K := x^*$.

Step 3. (Stopping criterion)

3.1 If $K < K_{max}$ *and* $f_{best} - H^K(x^*) > \epsilon$ *go to Step 1.*

For Lipschitz functions we can choose h^K at step 1.1 as

$$h^K(x) := f(x^K) - Cd_P\left(x, x^K\right), C \geq M.$$

The underestimate H^K given by

$$H^K(x) := \max_{k=1,\ldots,K}\left(f(x^k) - Cd_P\left(x, x^k\right)\right)$$

is often called the *saw-tooth* underestimate, or saw-tooth cover of f, because of its shape (Fig. 1). The minimization problem solved at Step 2 is called the relaxed problem.

By choosing the simplicial distance function (5) the relaxed problem takes the form

$$\begin{aligned} \text{minimize} \quad & \max_{k=1,\ldots,K} \min_{i=1,\ldots,n+1} \left(f(x^k) - C\left(x_i^k - x_i\right)\right) \\ \text{s.t.} \quad & x \in S', \end{aligned} \tag{11}$$

where S' is the feasible domain in the plane $\{x \in R^{n+1} | \sum x_i = 1\}$ [we remind about the slack variable in (5)].

The point in using the simplicial distance d_P rather than any norm is that the relaxed problem has a special structure facilitating a particularly efficient algorithm for its solution.

3.1 Solution to the relaxed problem for ECAM

This section reproduces details of the Cutting Angle algorithm from [1,2,4–6]. These details are essential for understanding how a modification of ECAM for quasiconvex functions works.

Let us define the *support vectors* l^k, $k = 1, 2, \ldots$:

$$l_i^k := \frac{f(x^k)}{C} - x_i^k, \quad i = 1, \ldots, n+1. \tag{12}$$

As the support functions, we use

$$h^k(x) := \min_{i=1,\ldots,n+1} \left(f(x^k) - C\left(x_i^k - x_i\right)\right) = \min_{i=1,\ldots,n+1} C\left(l_i^k + x_i\right). \tag{13}$$

We are interested in locating all local minima of the function

$$H^K(x) := \max_{k=1,\ldots,K} h^k(x) \tag{14}$$

in S', which after sorting will yield the global minimum of H^K.

Theorem 3.1 [5,6] *A necessary and sufficient condition for a point $x^* \in ri\ S'$ to be a local minimizer of $H^K(x)$ given by (13),(14) is that there exist an index set $J = \{k_1, k_2, \ldots, k_{n+1}\}$ of cardinality $n+1$, such that*

$$d = H^K(x^*) = C\left(l_1^{k_1} + x_1^*\right) = C\left(l_2^{k_2} + x_2^*\right) = \cdots = C\left(l_n^{k_{n+1}} + x_{n+1}^*\right),$$

and $\forall i \in \{1, \ldots, n+1\}$,

$$\left(l_i^{k_i} + x_i^*\right) < \left(l_j^{k_i} + x_j^*\right),\ j \neq i.$$

Let x^* be a local minimizer of $H^K(x)$, which corresponds to some index set J satisfying conditions of Theorem 3.1. Form the ordered combination of the support vectors $L = \{l^{k_1}, l^{k_2}, \ldots, l^{k_{n+1}}\}$ that corresponds to J. It is helpful to represent this combination with a matrix L whose rows are the support vectors l^{k_i}:

$$L := \begin{pmatrix} l_1^{k_1} & l_2^{k_1} & \cdots & l_{n+1}^{k_1} \\ l_1^{k_2} & l_2^{k_2} & \cdots & l_{n+1}^{k_2} \\ \vdots & \vdots & \ddots & \vdots \\ l_1^{k_{n+1}} & l_2^{k_{n+1}} & \cdots & l_{n+1}^{k_{n+1}} \end{pmatrix}, \tag{15}$$

so that its components are given by $L_{ij} = \frac{f(x^{k_i})}{C} - x_j^{k_i}$.

Theorem 3.2 [6] *Let the support vectors $l^k, k = 1, \ldots, K$ be defined using (12). Let x^* denote a local minimizer of $H^K(x)$ in $ri\ S'$ and $d = H^K(x^*)$. Then matrix (15) corresponding to x^* enjoys the following properties:*

(1) $\forall i, j \in \{1, \ldots, n+1\}, i \neq j : l_i^{k_j} > l_i^{k_i}$;
(2) $\forall r \notin \{k_1, k_2, \ldots, k_{n+1}\}\ \exists i \in \{1, \ldots, n+1\} : L_{ii} = l_i^{k_i} \geq l_i^r$;
(3) $d = \frac{Trace(L)+1}{C^-}$, *and*

(4) $x_i^* = \frac{d}{C} - l_i^{k_i}, i = 1, \ldots, n+1$, *where* $C^- = \sum_{i \in \{1,\ldots,n+1\}} \frac{1}{C} = \frac{n+1}{C}$.

Condition (1) of the Theorem 4 reads that the diagonal elements of matrix L are *dominated* by their respective columns, and condition (2) reads that no support vector l^r (which is not part of (L) strictly dominates the diagonal of L. The approach taken in [2,3] is to enumerate all combinations L with the properties (1), (2), which will give the positions of local minima x^* and their values d by using (3)–(4).

Furthermore, combinations L determine a partition of S' into subsets $A(L)$, such that in the interior of each subset the local minimum d of H^K is unique. The following system of inequalities determines a subset $A(L)$

$$\forall i, j \in \{1, \ldots, n+1\}, i \neq j : C\left(x_j - x_j^{k_j}\right) \leq C\left(x_i - x_i^{k_j}\right), \tag{16}$$

and on each $A(L)$ function H^K is given by

$$H^K(x) = \max_{i=1,\ldots,n+1} C\left(x_i + l_i^{k_i}\right). \tag{17}$$

In [2,3] we showed that combinations L can be built incrementally, by taking initially the first $n+1$ support vectors (which yields the unique combination $L = \{l^1, l^2, \ldots, l^{n+1}\}$), and then adding one new support vector at a time. Clearly,

$$H^k(x) = \max\{H^{k-1}(x), h^k(x)\}, k = n+2, \ldots, K. \tag{18}$$

Suppose, we have already identified the local minima of $H^{k-1}(x)$, i.e., all the required combinations L. Let us denote this set by V^{k-1}. When we add another support vector l^k, we can "inherit" most of the local minima of $H^{k-1}(x)$ (a few will be lost since condition (2) of Theorem 4 may fail with l^k playing the role of l^r), and we only need to add a few new local minima, that are new combinations L necessarily involving l^k.

We proved in [2] that these new combinations are simple modifications of those combinations that were discarded because they failed (2) with $l^r = l^k$. Furthermore, in [3,6] we proved that all local minimizers of the functions $H^{n+1}, \ldots, H^k, \ldots H^K$ can be represented in a tree structure, in which the required minima of H^K are the leaves. The root of the tree is $V^{n+1} = \{(l^1, l^2, \ldots, l^{n+1})\}$. It is possible to perform test of the condition (2) by starting the test from the root using depth-first search. This results in a highly efficient algorithm for enumeration of local minima of H^K [3,4,6].

This algorithm consists of two parts. The inner recursive part updates the tree of local minima T^{K-1} to T^K. The main Algorithm 3.3 calls the Algorithm 3.2 to maintain the tree T^K. A special data structure was designed to hold the information about the local minimizers of H^K [7]. It consists of an $n+1$-ary tree and a priority queue (binary heap) to hold the references to the leaves of the tree. The use of the priority queue simplifies sorting out local minima of H^K, as locating the global minimum is $O(1)$ operation. The maintenance of the heap at every iteration takes $O(\log |V^K|)$ operations.

Algorithm 3.2 ***Update of the tree*** T^{K-1}
Input: The tree T^{K-1} *of local minima of* $H^{n+1}, H^{n+2}, \ldots, H^{K-1}$*; the new support vector* l^K*; tested node* L.
Output: The tree T^K*; the set of leaves* V^K.

Step 1 Test L against condition (2), with $l^r = l^K$.
Step 2 If test succeeds, go to Step 5 (cut off this branch).

Step 3 *If test fails, and L is not a leaf, then call Algorithm 3.2 (T^{K-1}, l^K, child (L), T^K, V^K) for all children of L. Go to Step 5*

Step 4 *Otherwise (test failed, and L is a leaf) add $n+1$ children to L. Each child node is a copy of L, with l^{k_i} replaced with l^K in the i-th child. Test condition (1) for each child. If test fails, delete this child node.*

Step 5 *If L is V^n (root), then $T^K = T^{K-1}$; $V^K = leaves(T^K)$ (we need to check this only once, at the first level of recursion). Return.*

Algorithm 3.3 ***Extended cutting angle algorithm***

T^K denotes the tree of local minima of functions $H^{n+1}, \dots, H^K$, V^K denotes the priority queue containing the leaves of the tree arranged in the order of increasing $d(L)$. $d(L)$ is computed by using property (3) *in Theorem 3.2.*

Step 0. (Initialisation)

0.1 *Take the initial points $x^k, k = 1, \dots, n+1$, and construct the support vectors $l^k, k = 1, \dots, n+1$ according to (12).*

0.2 *Set $K := n+1$, $L_{root} := \{l^1, l^2, \dots, l^{n+1}\}$, $T^{n+1} = V^{n+1} := \{L_{root}\}$.*

0.3 *$f_{best} := \min_{k=1,\dots,n+1} f(x^k)$.*

Step 1. (Form a new support vector)

1.1 *Choose $L^* := Head(V^K)$ (corresponds to the global minimum of H^K).*

1.2 *Form $x^* := x^*(L^*)$ using condition (4) of Theorem 3.2.*

1.3 *Evaluate $f^* := f(x^*)$. $f_{best} := \min\{f_{best}, f^*\}$.*

1.4 *Set $K := K+1$.*

1.5 *Form l^K using $l_i^K := \frac{f(x^*)}{C} - x_i^*$.*

Step 2. (Update T^K)

2.1 *Call Algorithm 3.2 ($T^{K-1}, l^K, V^{n+1}, T^K, V^K$).*

2.2 *Delete from V^K elements L if $d_P(L) < f_{best}$.*

Step 3. (Stopping criterion)

3.1 *If $K < K_{max}$ and $f_{best} - d_P(L^*) > \epsilon$ go to Step 1.*

4 Extended cutting angle method for quasiconvex functions (QECAM)

4.1 General scheme

Algorithm 3.3 can be applied directly to any *blsd* function, however, it does not take into account quasiconvexity. In this section we show how this property can be taken into account to achieve a significant improvement in the numerical performance of the algorithm. We consider the case when a subgradient is available.

If a subgradient is available, at those points where it is not zero we can determine the half-space which does not contain the minimum of f. Its intersection with the feasible domain S is given as

$$P^k := \{x \in S \mid \langle x - x^k,\ p^k \rangle \geq 0\},$$

where $p^k \in \partial f(x^k)$ (or $p^k = \nabla f(x^k)$ if the gradient is available). Thus for all $x \in P^k$: $f(x) \geq f(x^k)$. Because of the Lipschitz condition, the function

$$g^k(x) := f(x^k) - M'd'(x, P^k) = f(x^k) - M' \min_{z \in P^k} d'(x, z)$$

is an underestimate of f. d' needs not coincide with d, in which case the Lipschitz constants M and M' differ (although M' can be calculated from M and vice versa, because of the equivalence of the Minkowki gauge and norms, see property (3) of Minkowki gauges in Sect. 2.1). If d' is an inner product based distance, we will have

$$g^k(x) = \min \left\{ f(x^k),\ f(x^k) + \langle x - x^k,\ p^k \rangle \right\} \tag{19}$$

(see Eq.(8)). Therefore we can improve the saw-tooth underestimate in (13), (14) by using

$$H_q^K(x) := \max_{k=1,\ldots,K} \left\{ h^k(x),\ f(x^k) - M'd'(x, P^k) \right\}. \tag{20}$$

Note that at those points x^k where the gradient is zero (or not available), $P^k = \emptyset$ and we put $d'(x, \emptyset) := -\infty$.

The underestimate (20) is applicable to functions with flat parts, in which case the subgradient does not help to improve the underestimate (13),(14), and the Algorithm 3.1 proceeds as with general Lipschitz functions. But at those points where the subgradient is not zero, this information improves the underestimate, and results in a faster convergence of the algorithm. We discuss its use in the next section.

The Algorithm 3.1 requires the following changes: the function H^K is replaced with H_q^K at Steps 1.2, 2.1 and 3.1. This is a formal change, however, it drastically affects the structure of the relaxed problem at Step 2.1. The purpose of the next sections is to show how to incorporate these changes while preserving the efficiency of the Algorithm 3.3.

4.2 Fathoming

Algorithm 3.3 processes the local minimizers of the functions H^K, $K = 1, 2, \ldots$ to determine the global minimum of H^K needed at Step 2.1. The list of local minimizers is organized into a priority queue, so that at each iteration the global minimum is at the top.

When we use the improved underestimate H_q^K, we can firstly verify whether the global minimum x^* of H^K is also the minimum of H_q^K. A straightforward way to do this is to compare $H^K(x^*)$ with $\max_{k=1,\ldots,K}\{f(x^k) - M'd'(x^*, P^k)\}$.

Next, if we established that x^* is not a global minimum of H_q^K, we can move this minimizer in the queue according to the value $H_q^K(x^*)$, without evaluating $f(x^*)$, and proceed with the steps of Algorithm 3.3. However, we can do even better and eliminate the whole subset $A(L)$ which corresponds to x^*. This step is known as fathoming. What this means is that we can cut the whole branches of the tree T^K, and significantly speed up the algorithm and reduce storage requirements.

The reason why $A(L)$ can be eliminated is that for all K, we have $H^K \leq H^{K+1}$, and also $H^K \leq H_q^K \leq H_q^{K+1}$. Then if we establish that the minimum of H_q^K on $A(L)$ is larger than f_{best}, then

$$f_{best} < \min_{x \in A(L)} H_q^K(x) \leq \min_{x \in A(L)} H_q^m(x) \text{ for } m > K.$$

Thus, the global minimum of f cannot be achieved on $A(L)$, and this subset can be fathomed.

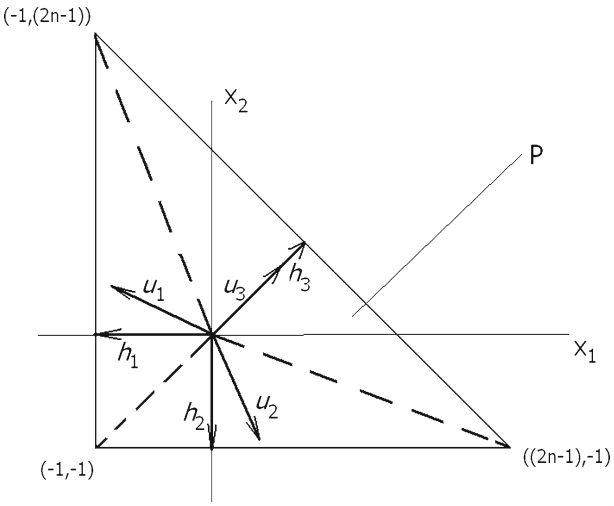

Fig. 2 The simplicial distance defined by (4), and the unit vectors u_i in (24)

However, by itself this does not imply that the node L can be eliminated from the tree containing the minimizers of H^K, which is the key to numerical efficiency. By Proposition 6 in [3], all local minimizers of $H^{K+1}, \ldots$, that are descendants of L will necessarily be confined to the subset $A(L)$. This is the main reason why elimination of the node L will not result in fathoming a superset of $A(L)$. Hence, elimination of the node L is equivalent to fathoming the subset $A(L)$.

Thus our strategy will be to use underestimates (19) to compute the lower bounds on $H_q^K(x)$ on the subsets $A(L)$. This will involve a bi-level mathematical programming problem

$$\begin{aligned} \text{minimize } H_q^K(x) = \max \left\{ \max_{i=1,\ldots,n+1} C\left(x_i + l_i^{k_i}\right), \max_{k=1,\ldots,K} \left\{ f(x^k) - M' d'(x, P^k) \right\} \right\} \\ \text{s.t. } x \in A(L) \cap S', \end{aligned} \tag{21}$$

where as earlier, d' is any inner product distance, in particular we use the Euclidean norm, and M' is the Lipschitz constant in this norm. [Note that if one knows M' in the Euclidean norm, one can calculate M in the simplicial distance by $M = \sqrt{n(4n-3)}M'$, and if one knows M, one can take $M' = M$, see Fig. 2 and Eq. (24) below].

The solution of (21) provides us with the exact lower bound $d_q(A)$ on f on the subset $A(L)$, which we can use to either eliminate $A(L)$ completely, if $d_q(A) > f_{\text{best}}$, or move the minimum of $d_q(A)$ in the priority queue according to its value. However, solution to (21) is numerically expensive,as K could be quite large. In what follows, we study possible shortcuts which provide us with less accurate but easily computable lower bounds on f.

4.3 Bounds on the relaxed problem for QECAM

The first shortcut is to consider the bounds given by $d_q(A)$ only if $x^* \in P^k$ for some k. To establish whether $x \in P^k$ we only need the sign of the dot product $\langle x - x^k,\ p^k \rangle$.

Secondly, we notice that from $\min_{x\in A}\max_{k=1,\ldots,K} f_k(x) \geq \max_{k=1,\ldots,K}\min_{x\in A} f_k(x)$ and (21) it follows that

$$\min_{x\in A(L)\cap S'} H_q^K(x) \geq \max_{k=1,\ldots,K} g^k(x) \tag{22}$$

$$\text{where } g^k(x) = \min_{x\in A(L)\cap S'} \max\left\{\max_{i=1,\ldots,n+1} C\left(x_i + l_i^{k_i}\right), f(x^k) - M'd'(x, P^k)\right\}$$

Then if we have $f_{\text{best}} \leq \max_{k=1,\ldots,K} g^k(x) \leq \min_{x\in A(L)\cap S'} H_q^K(x)$, the node L can be fathomed. Consider for a fixed k the problem

$$\begin{aligned}\text{minimize } g^k(x) = \max&\left\{\max_{i=1,\ldots,n+1} C\left(x_i + l_i^{k_i}\right), f(x^k) - M'd'(x, P^k)\right\}\\ \text{s.t. } x &\in A(L)\cap S'.\end{aligned} \tag{23}$$

Let g^{k*} denote its solution. The maximum $\max_{k=1,\ldots,K} g^{k*}$ is a lower bound on f on $A(L)$, and if $f_{\text{best}} \leq \max_{k=1,\ldots,K} g^{k*}$, the note L can be fathomed.

While the problem (23) can be converted to a small scale linear programming problem, it is the large number of such small problems that would make the algorithm inefficient. Let us consider an underestimate

$$g^k_{\text{under}} := \min_{x\in R} g^k(x) \leq \min_{x\in A(L)\cap S'} g^k(x),$$

where $R := \cup_{i=1}^{n+1} R_i \subset B(L)$ consists of segments of straight lines

$$R_i := \left\{x \in B(L) |\ \forall j, m \neq i :\ C\left(x_j + l_j^{k_j}\right) = C\left(x_m + l_m^{k_m}\right)\right\},$$

and $B(L) \supset A(L)$ is defined as

$$B(L) := \left\{x \in R^{n+1} |\ x_i \leq x_i^{k_i}, i = 1, \ldots, n+1\right\}.$$

We note that

$$\min_{x\in R}\max_{i=1,\ldots,n+1} C\left(x_i + l_i^{k_i}\right) \leq \min_{x\in A(L)}\max_{i=1,\ldots,n+1} C\left(x_i + l_i^{k_i}\right).$$

Then if $g^k_{\text{under}} = \min_{x\in R} g^k(x)$ is achieved on at some x where $\max_{i=1,\ldots,n+1} C\left(x_i + l_i^{k_i}\right) = f(x^k) - M'd'(x, P^k)$, g^k_{under} is smaller than $\min_{x\in A(L)\cap S'} g^k(x)$. On the other hand, if $f(x^k) - M'd'(x, P^k) > \max_{i=1,\ldots,n+1} C\left(x_i + l_i^{k_i}\right)$ for all $x \in R$, then the same inequality holds for $x \in A(L)$. Hence g^k_{under} is indeed an underestimate, which in addition is easily computed.

Such subsets are illustrated on Fig. 3. The graph of the function $H^k(x)$ on $A(L)$ consists of $n+1$ planes intersecting at (x^*, d), and we extend that function to $B(L)$. The subsets R_i are the abscissae of points where all but the i-th plane intersect. They are segments of straight lines between x^* and the boundary of $B(L)$, with the directional vector u_i, given by (see Fig. 2)

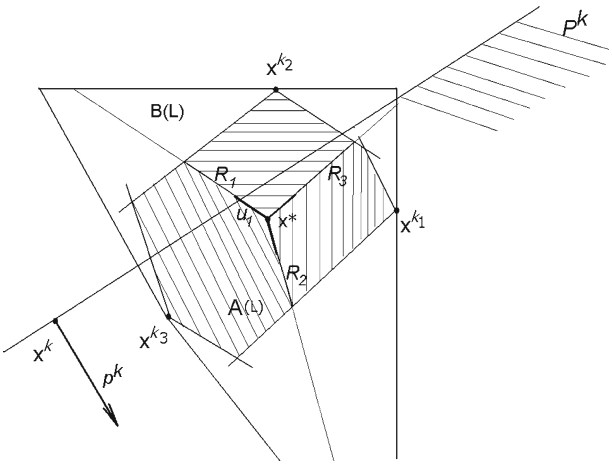

Fig. 3 The set $A(L)$, plane P^k, subsets R_i and the level curves of the function H^K in $A(L)$ given by (17)

$$\begin{aligned} u_1 &:= \frac{1}{\sqrt{n(4n-3)}}(-(2n-1), 1, 1, \ldots, 1), \\ u_2 &:= \frac{1}{\sqrt{n(4n-3)}}(1, -(2n-1), 1, \ldots, 1), \\ &\ldots \\ u_n &:= \frac{1}{\sqrt{n(4n-3)}}(1, 1, \ldots, 1, -(2n-1)), \\ u_{n+1} &:= \frac{1}{\sqrt{n}}(1, \ldots, 1). \end{aligned} \tag{24}$$

Moreover, when restricted to R_i, the function $H^k(x)$ is given by

$$H^k_{R_i}(t) = d + C'_i t, \quad 0 \le t \le T_i,$$

with $C'_i := \frac{C}{\sqrt{n(4n-3)}}$, $i = 1, \ldots, n$, $C'_{n+1} := \frac{C}{\sqrt{n}}$ and T_i is determined from the point of intersection of R_i and the boundary of $B(L)$ [for computational purposes we can use an equivalent simpler condition $H^k_{R_i}(t) \le \min_{j \ne i} f(x^{k_j})$].

Let us take the Euclidean distance as d'. Now for each subset R_i we have a univariate problem

$$\min_{x \in R_i} g^k(x) = \min_{0 \le t \le T_i} \max \left\{ d + C'_i t,\, f(x^k) - M''(t - D) \right\}, \tag{25}$$

where $M'' := \frac{<u_i,\, p^k>}{||p^k||} M'$ and $D := -\frac{<x^* - x^k,\, p^k>}{<u_i,\, p^k>} > 0$. The solution is given as

$$g_i^{k*} = \max \left\{ f(x^k) - M''(T_i - D), \frac{M'' d + C'_i \left(M'' D - f(x^k) \right)}{M'' + C'_i} \right\}. \tag{26}$$

Figure 4 illustrates this point.

Now, to calculate the overall underestimate we take

$$g := \max_{k \in K'} g^k_{\text{under}} = \max_{k \in K'} \min_{i=1,\ldots,n+1} g_i^{k*} \le \max_{k \in K'} g^{k*}, \tag{27}$$

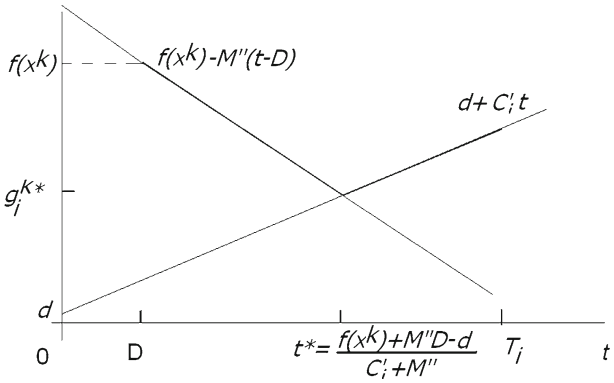

Fig. 4 Solution to problem (25)

where K' is the set of indices, such that $\langle x^* - x^k,\ p^k\rangle \geq 0$ for $k \in K'$. Then, if $f_{\text{best}} \leq g$, we perform fathoming and eliminate the node L.

The advantage of using the underestimate g of the bound $H_q^k(x)$ on $A(L)$ is that it is easily computed using an explicit formula. The computational complexity is $O(Kn)$, and several quantities can be precomputed. On the other hand our numerical experiments have shown little difference between g and the actual minimum of $H_q^K(x)$ on $A(L)$.

The above considerations give rise to the following algorithm. Note that because of the efficiency of the Algorithm 3.3 in processing the minima of H^K, we use formally H^K (and not H_q^K) in the Algorithm 4.1 below, however, we perform the fathoming step (Step 1.3) using the underestimates of H_q^K, g, so effectively we use H_q^K without computing them explicitly.

Algorithm 4.1 ***Extended cutting angle algorithm for quasiconvex functions (QECAM)***
The algorithm assumes f is quasiconvex, and $p \in \partial f(x)$ can be computed (not necessarily for all x). The set K' denotes the set of indices k, such that $p^k \in \partial f(x^k)$ is available and it is not 0.
T^K denotes the tree of local minima of functions $H^{n+1}, \ldots, H^K$, V^K denotes the priority queue containing the leaves of the tree arranged in the order of increasing $d_P(L)$. $d = d_P(L)$ is computed by using property (3) *in Theorem* 3.2.

Step 0. (Initialization)

0.1 Take the initial points $x^k, k = 1, \ldots, n+1$, and construct the support vectors $l^k, k = 1, \ldots, n+1$ according to (12). *Record the subgradients $p^k \in \partial f(x^k)$, if available and maintain K'.*
0.2 Set $K := n+1$, $L_{root} := \{l^1, l^2, \ldots, l^{n+1}\}$, $T^{n+1} = V^{n+1} := \{L_{root}\}$.
0.3 $f_{best} := \min_{k=1,\ldots,n+1} f(x^k)$.

Step 1. (Form a new support vector)

1.1 Choose $L^* := Head(V^K)$ (corresponds to the global minimum of H^K).
1.2 Form $x^* := x^*(L^*)$ using condition 4) *of Theorem* 3.2.
1.3 Obtain an improved bound on f using quasiconvexity.

1.3.1 Determine the subset $A(L)$ from (16).

1.3.2 For all $k \in K'$ *do*
For all $i = 1, \ldots, n+1$ *determine the quantities* D, M'' *in* (25).
For all $i = 1, \ldots, n+1$ *compute* g_i^{k*} *using* (26).
Compute $g_{under}^k := \min_{i=1,\ldots,n+1} g_i^{k*}$.
1.3.3 Compute $g := \max_{k \in K'} g_{under}^k$.
Let x^{**} *denote the minimizer corresponding to* g.
1.3.4 Fathoming: if $f_{best} \leq g$ *then delete node* L *and go to Step* 1.
1.3.5 Otherwise let $x^* := x^{**}$.

1.4 Evaluate $f^* := f(x^*)$. $f_{best} := \min\{f_{best}, f^*\}$.
1.5 Evaluate $p^k \in \partial f(x^k)$ *if possible. If it is, and* $p^k \neq 0$, *augment* K'.
1.6 Set $K := K + 1$.
1.7 Form l^K *using* $l_i^K := \frac{f(x^*)}{C} - x_i^*$.

Step 2. (Update T^K*)*

2.1 Call Algorithm 3.2 *(*$T^{K-1}, l^K, V^{n+1}, T^K, V^K$*).*
2.2 Delete from V^K *elements* L *if* $d_P(L) < f_{best}$.

Step 3. (Stopping criterion)

3.1 If $K < K_{max}$ *and* $f_{best} - d_P(L^*) > \epsilon$ *go to Step* 1.

Notes. (1) The unit vectors u_i in (24) that are used to compute D, M'' at Step 1.3.2, as well as constants C_i', are precomputed. (2) The values $||p^k||$ and $\langle u_i, p^k \rangle$ are computed once at step 1.5 and saved in a lookup table.

5 Results of numerical experiments

We illustrate the performance of the new algorithm on special instances of quasiconvex problems: fractional programming and location problems. We do not intend to compare our method with specialized methods designed for these particular instances, since naturally, such specialized methods take into account the specifics of the problem, and more general methods do not. On the other hand, for general quasiconvex programs, the existing methods may fail to converge in some instances because of the flat parts of the objective function.

The optimal performance of economic systems is often formulated as the maximization or minimization of a ratio of two functions (sometimes the objective function involves more than one such ratio)

$$\inf \{p(x)/q(x) \mid x \in D\}, \tag{28}$$

where p and q are real-valued functions with p positive, defined on a nonempty set X, $X \subset \mathbb{R}^n$, $D := \{x \in X \mid h(x) \leq 0\}$ and $h : X \mapsto \mathbb{R}^m$. In order to have a quasiconvex program we require that X be convex, p convex and q concave (or p nonnegative and q affine) and the functions h_j, $j = 1, \ldots, m$ to be convex. The following objective functions frequently occur: maximization of productivity, maximization of return on investment, maximization of return on risk, maximization of output on input and minimization of cost on time (see [16,21]). There is a rich class of algorithms based in the Dinkelbach method (see [13] and references in [19]), in which the solution is obtained by solving a parametric programming

Table 1 Numerical results for fractional problems

$\epsilon = 0.01$		QECAM			ECAM		
n	m	Time	Iter found	Iter confirmed	Time	Iter found	Iter confirmed
2	2	0.03	221	261	55	7315	36817
	3	0.11	506	524	61	6616	40000
	5	0.14	462	605	45	7878	31940
	10	0.06	468	492	53	10844	35882
	25	0.07	365	479	34	8363	26606
3	2	287	5084	7694	600	–	80000
	3	617	12124	12758	600	53151	80000
	5	165	6056	7972	600	–	80000
	10	323	13604	16371	600	78312	80000
	25	189	13301	20102	600	–	80000
4	2	915	6058	15360	600	–	80000
	3	926	13498	16083	600	–	80000
	5	1042	8919	19144	600	–	80000
	10	681	6633	14630	600	–	80000
	25	823	7443	16141	600	–	80000

The average running time and the number of function evaluations are reported. *Iter found* is the number of objective function evaluations needed to find the global minimum, and *iter confirmed* is the total number of evaluations needed to confirm it. For $n > 2$, ECAM did not achieve the desired precision in the indicated time, although it has found a solution within 5% from the global minimum

problem. In [14], a generalization of the ellipsoid algorithm is developed to solve quasiconvex programs also with a convergency proof. The generalization of the ellipsoid method is applied to solve a class of quasiconvex location problems.

One drawback in the application of the above-mentioned algorithms comes from smooth and nonsmooth quasiconvex functions whose graph presents flat parts.

Two types of problems have been randomly generated. The results are described in Tables 1 and 2 in which ϵ is the given precision and every row corresponds to averages of 10 instances. Note that ϵ is the accuracy with which the method *guarantees* the globally optimal solution (i.e., the difference between the minimum found and its guaranteed underestimate), which is different to the accuracy compared to the known solution (because of the way test problems were generated, the true global minimum was unknown).

Under QECAM and ECAM headings we list under *time* the average time taken by the corresponding algorithm in seconds, and under *iter found* and *iter confirmed* the average total number of objective function evaluations before the algorithm finds and then confirms the global minimum.

All required *CPU* times reported refer to the computer used ACER TravelMate 8100 with 1 GB of *RAM* and a single Pentium IV *CPU* of 2.1 GHz.

Before we proceed with the discussion of numerical results, it is illustrative to look at a simple two-variate optimization problem and examine the implications the step 1.3 of the QECAM algorithm had on choosing the points x^k. Consider the following test function

$$f(x) = \max \left\{ \begin{array}{l} (x-1)^2 + (y-2)^2, \\ -1 + 0.15\sqrt{x^2 + 0.5y^2}, \\ -1.25 + 0.2\sqrt{x^2 + y^2}, \\ 0.2\sqrt{|0.5 - y|} - 1, \\ -0.7 - 2^{-(x-1)^2-(y-2)^2} \end{array} \right\}; \tag{29}$$

with global solution $a = (0.303\ldots, 0.683\ldots)$ on $S = [-3, 3]^2$. It is a quasiconvex function, and it has flat parts, as can be seen from its graph on Fig. 5. Figure 6

Table 2 Numerical results for location problems

$\epsilon = 0.01$		QECAM			ECAM		
m	p	Time	Iter found	Iter confirmed	Time	Iter found	Iter confirmed
5	1.1	0.8	1360	1988	43.1	15010	30970
	1.5	0.88	1560	2574	64	14243	40000
	1.9	0.95	2181	2649	64	12888	40000
	2.1	0.2	752	1098	52	5316	37670
	3.0	0.6	1735	1930	44	9681	53588
	Mix	0.6	1601	1981	64	12433	40000
25	1.1	1.4	2555	2751	33	5024	24610
	1.5	0.44	1202	1700	38	9298	28970
	1.9	0.27	980	1119	49	9730	33747
	2.1	0.41	1486	1706	55	15122	36065
	3.0	0.33	1040	1210	55	8240	35796
	Mix	0.25	940	1154	30	6844	23300
50	1.1	1.04	1643	2389	30	1784	25586
	1.5	0.5	1244	1552	37	3483	27577
	1.9	0.31	937	1290	23	3990	22975
	2.1	0.15	606	853	42	5489	29140
	3.0	0.2	868	958	36	6052	27770
	Mix	0.43	1320	1357	26	6212	21556
100	1.1	0.8	1504	1811	42	2348	27745
	1.5	0.1	491	504	28	5428	20823
	1.9	0.1	457	500	41	5539	27333
	2.1	0.1	421	425	28	5372	21138
	3.0	0.1	429	528	42	5427	27693
	Mix	0.5	1357	1696	26	3978	21630
250	1.1	0.9	1684	1941	9.5	2093	13322
	1.5	0.8	1711	1928	35	1449	25006
	1.9	0.4	1042	1320	10	1887	14325
	2.1	0.17	593	695	6.5	2544	10997
	3.0	0.25	641	889	31	2009	21731
	Mix	0.85	1430	1867	23	1342	18774

The average running time and number of function evaluations are reported. *Iter found* is the number of objective function evaluations needed to find the global minimum, and *iter confirmed* is the total number of evaluations needed to confirm it

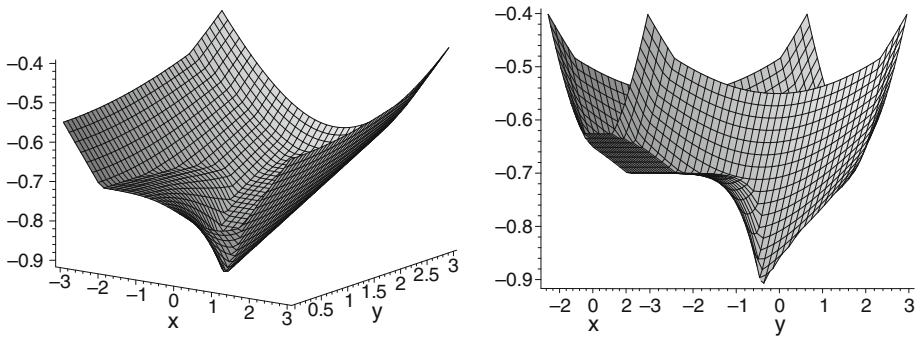

Fig. 5 Test function (29)

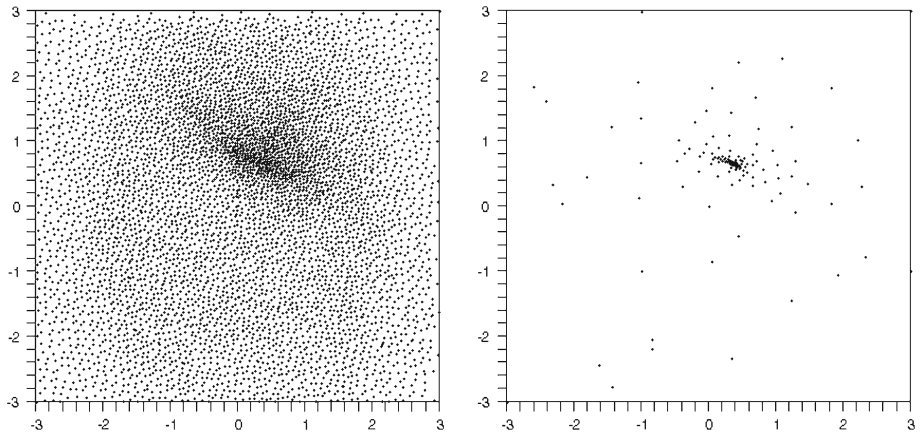

Fig. 6 The points x^k chosen by ECAM *(left)* and QECAM *(right)* when minimizing test function (29)

presents the scatterplots of the points x^k chosen by the ECAM and QECAM algorithms, respectively. We see that ECAM thoroughly samples the whole domain, and slowly converges to the global minimum (where the dots concentrate). In contrast, QECAM is more efficient in fathoming big parts of the domain by cutting a number of half-spaces.

5.1 Fractional programming

The fractional programming example considered in this article comes from Boncompte and Martínez-Legaz [10] and it is defined as follows:

- We consider instances of the objective function

$$f(x) = \max\{(a_i x + \alpha_i)/(b_i x + \beta_i),\ i = 1, \ldots, m\},$$

with $a_i,\ b_i \in I\!R^n$ and $\alpha,\ \beta \in I\!R$ and the additional assumption that the denominators are positive in the feasible domain D of $I\!R^n$ (compact and convex). This happens, e.g., when b_i's are nonnegative, the β_i's are positive and D is contained in the nonnegative orthant.
- With these assumptions the objective function is blsd on D.
- Instances have been generated with $n := 1, 2, 3, \ldots, k, \ldots, m := 1, 2, \ldots, k, \ldots$. The coefficients a_i and coefficients α_i are generated according to a uniform distribution $U[-50,\ 50]$, the coefficients b_i are generated using $U[0,\ 50]$ and the coefficients β_i are generated using $U[0.01,\ 50]$.

5.2 Location problems

The single facility location example comes from dos Santos Gromicho [14] and is defined as follows:

$$\begin{array}{l}\text{minimize} \quad f(x)\\ \text{subject to: } x \in I,\end{array} \tag{30}$$

where I is a closed hyperrectangle and

- $f(x) = \max\{f_i(x),\ i = 1, \ldots, m\}$,
- $f_i(x) := \varphi_i(\gamma(x))$ with

$$\varphi_i : I\!R^m_+ \mapsto I\!R,\ i = 1, \ldots, m,$$

nondecreasing and continuous functions on an open set A, $I\!R^m_+ \subset A$, so called *disutility functions*, and

$$\gamma : I\!R^n \mapsto I\!R^m_+$$

defined in the form $\gamma(x) := (\gamma_1(x - d_1), \ldots, \gamma_m(x - d_m))$ with $\gamma_i = \gamma_{G_i}$ a *Minkowski gauge*, where each G_i is a compact convex set of $I\!R^n$ and $0 \in \operatorname{int} G_i, i = 1, \ldots, m$.

We know that:

- If $\varphi_i,\ i = 1, \ldots, m$ are nondecreasing and quasiconvex on an open set A, $I\!R^m_+ \subset A$, then the function f_i is quasiconvex. Therefore the objective function f is quasiconvex as well.
- If $\varphi_i,\ i = 1, \ldots, m$ are nondecreasing and differentiable and quasiconvex on an open set A, $I\!R^m_+ \subset A$ and their gradients are locally bounded, then the objective function f is boundedly lower subdifferentiable for every compact set of $I\!R^n$.

We take the following parameters:

- Dimension $n = 2$.
- The number of demand points $d_i,\ i = 1, \ldots, m$, with $m \in \{5,\ 25,\ 50,\ 100,\ 250\}$.
- All the demand points are generated within $I := [0,\ 250] \times [0,\ 250]$.
- The weight w_i of each demand point d_i is determined as follows. Let $\bar{w}_i, i = 1, \ldots, m$ be drawn from $U(0,\ 1]$ then set $w_i = \bar{w}_i / \sum_{j=1}^m \bar{w}_j, i = 1, \ldots, m$. Notice that the w_i are not uniformly distributed on the unit simplex and $\sum_{i=1}^m w_i = 1$.
- For each demand point d_i the φ_i function is defined,

$$\varphi_i(z) = \begin{cases} 100 w_i \log(z_i + 1), & \text{for } 1 \le i \le \lceil m/3 \rceil,\\ 100 w_i \arctan(z_i), & \text{for } \lceil m/3 \rceil + 1 \le i \le \lceil 2m/3 \rceil,\\ 5 w_i z_i, & \text{for } \lceil 2m/3 \rceil + 1 \le i \le m,\end{cases}$$

with $\lceil x \rceil$ the ceiling of $x \in I\!R$. Notice that to prevent f to become convex we assign to the concave part of the functions φ_i a bigger weight than to the linear part.

- The Minkowski gauge is

$$\gamma_i(x) = \|x\|_{p_i},$$

with $p_i \in \mathcal{P} = \{1.1, 1.5, 1.9, 2.1, 3.0\}$. Then,

$$\gamma(x) = \left(\|x - d_1\|_{p_{k_1}}, \ldots, \|x - d_m\|_{p_{k_m}}\right),$$

with $p_{k_m} \in \mathcal{P}$. Two different classes of sample problems can be constructed,

(a) $p_{k_1} = p_{k_2} = \cdots = p_{k_m} = p \in \mathcal{P}$,
(b) every p_{k_i}, $i = 1, \ldots, m$, is randomly selected from $\mathcal{P}$ (mixed sample).

- The constant

$$\sqrt{2}\max\{\max\{100w_i,\ i = 1, \ldots, \lceil 2m/3\rceil\},\ \max\{5w_i,\ i = \lceil 2m/3\rceil + 1, \ldots, m\}\}$$

is a Lipschitz constant for the function f (for $n = 2$).

6 Discussion and concluding remarks

The numerical experiments confirm the superior performance of QECAM for problems with quasiconvex Lipschitz objective functions. The gain in efficiency is quite remarkable for two-variate problems (see Table 2). Interestingly, ECAM usually locates the global minimum quite early, but spends much time confirming it. QECAM locates the global minimum early and confirms it soon after, which is due to the efficiency of fathoming.

For $n > 2$, ECAM frequently failed to locate the global minimum, even though its output was within 5% of the solution. QECAM did locate and confirm the global minimum, although its running time was significantly bigger than for two-dimensional problems. This is not surprising given NP-hardness of the global optimization problem, and the fact that the number of local minima of the underestimate H^k grows exponentially with $\lceil \frac{n}{2} \rceil$.

We developed a modified version of the Extended Cutting Angle Method, designed for quasi-convex programming. We know that ECAM algorithm can be applied directly to any *lbsd* function, however, it does not take into account quasiconvexity which can improve its numerical performance. The new algorithm QECAM takes into account quasiconvexity of the objective function to efficiently fathom big parts of feasible domain. QECAM inherits numerical efficiency of ECAM, which is based on a representation of the partition of the feasible domain and of a piecewise linear underestimate of f using a tree structure.

While quasiconvex functions have nice structure and unique global minimum, their minimization is challenging because they may have flat parts, where other methods stop. QECAM is applicable to quasiconvex objective functions with flat parts. This is a significant improvement over existing methods. Numerical experiments show that using the quasiconvexity property in the ECAM significantly reduces the number of objective function evaluations and the total computing time.

Mathematical and computational modeling in the area of sustainable development frequently requires solving optimization problems of quasiconvex programming. Two examples of such problems, fractional programming and location problems were presented and investigated numerically.

Acknowledgements We would like to acknowledge the fundamental contributions of Professor Alexander M. Rubinov, who has recently passed away, in the fields of abstract convex and non-smooth analysis and

optimization. Alexander M. Rubinov was a researcher of highest academic excellence, who was admired by everyone for the breadth and depth of his knowledge and his readiness to share it. His work had great impact in several areas of mathematics and optimization (including this paper), and has laid solid fundamentals for future research as well. Alexander M. Rubinov was a fantastic example of great kindness, modesty and wisdom for all people who knew him.

References

1. Bagirov, A., Rubinov, A.M.: Modified versions of the cutting angle method. In: Hadjisavvas, N., Pardalos, P.M. (eds.) Convex Analysis and Global Optimization, Nonconvex Optimization and its Applications, Vol. 54, pp. 245–268. Kluwer, Dordrecht (2001)
2. Batten, L.M., Beliakov, G.: Fast algorithm for the cutting angle method of global optimization. J. Global Opt. **24**, 149–161 (2002)
3. Beliakov, G.: Geometry and combinatorics of the cutting angle method. Optimization **52**(4–5), 379–394 (2003)
4. Beliakov, G.: Cutting angle method—a tool for constrained global optimization. Optim. Methods Softw. **19**, 137–151 (2004)
5. Beliakov, G.: A review of applications of the cutting angle methods. In: Rubinov, A.M., Jeyakumar, V. (eds.) Continuous Optimization, pp. 209–248. Springer, New York (2005)
6. Beliakov, G.: Extended cutting angle method of global optimization. Paci. J. Optim. **4**, 153–176 (2008)
7. Beliakov, G., Ting, K.M., Murshed, M., Rubinov, A.M., Bertoli, M.: Efficient serial and parallel implementations of the cutting angle method. In: Di Pillo, G. (ed.) High Performance Algorithms and Software for Nonlinear Optimization, pp. 57–74. Kluwer, Norwell (2003)
8. Berman, O., Krass, D. (eds.): Recent Developments in the Theory and Applications of Location Models Part I. Kluwer, Dordrecht (2002a)
9. Berman, O., Krass, D. (eds.): Recent Developments in the Theory and Applications of Location Models Part II. Kluwer, Dordrecht (2002b)
10. Boncompte, M., Martínez-Legaz, J.E.: Fractional programming by lower subdifferentiability techniques. J. Optim. Theory Appl. **68**(1), 95–116 (1991)
11. Cheney, E.W., Goldstein, A.A.: Newton's method for convex programming and Tchebycheff approximation. Numer. Math. **1**, 253–268 (1959)
12. Demyanov, V.F., Rubinov, A.M.: Constructive Nonsmooth Analysis. Peter Lang, Frankfurt am Main (1995)
13. Dinkelback, W.: On nonlinear fractional programming. Manag. Sci. **13**, 492–498 (1967)
14. dos Santos Gromicho, J.A.: Quasiconvex Optimization and Location Theory, Applied Optimization, Vol. 9. Kluwer, Dordrecht (1998)
15. Hadjisavvas, N., Komlósi, S., Schaible, S. (eds.): Handbook of Generalized Convexity and Generalized Monotonicity, Volume 76 of Nonconvex Optimization and its Applications. Springer, New York (2005)
16. Horst, R., Pardalos, P.M. (eds.): Handbook of Global Optimization, Nonconvex Optimization and its Applications, Vol. 2. Kluwer, Dordrecht (1995)
17. Kelley, J.E. Jr..: The cutting-plane method for solving convex programs. J. Soc. Indust. Appl. Math. **8**, 703–712 (1960)
18. Plastria, F.: Lower subdifferentiable functions and their minimization by cutting planes. J. Optim. Theory Appl. **46**(1), 37–53 (1985)
19. Roubi, A.: Method of centers for generalized fractional programming. J. Optim. Theory Appl. **107**(1), 123–143 (2000)
20. Rubinov, A.M.: Abstract Convexity and Global Optimization, Nonconvex Optimization and its Applications, Vol. 44. Kluwer, Dordrecht; Boston (2000)
21. Schaible, S., Shi, J.: Recent developments in fractional programming: single-ratio and max- min case. In: Nonlinear Analysis and Convex Analysis, pp. 493–506. Yokohama, Yokohama (2004)