Journal of Global Optimization **24:** 115–132, 2002.

Solving Problems with Min-Type Functions by Disjunctive Programming

MIKHAIL ANDRAMONOV
Institute of Mathematics and Mechanics, Universitetskaya St 17, 420008, Kazan, Russia

Abstract. We consider applications of disjunctive programming to global optimization and problems with equilibrium constraints. We propose a modification of the algorithm of F. Beaumont for disjunctive programming problems and show its numerical efficiency.

Key words: Disjunctive programming; Global optimization; Surrogate constraints

1. Introduction

Global optimization is a popular and important field of research. In general, the problems of global optimization are very difficult to solve due to their combinatorial nature (see [14, 15]), however, it is possible to solve specially structured problems efficiently. For example, there exist a number of methods of minimizing Lipschitz functions, which are a very important class of functions [23]. Recently one of the important tools for global optimization has become abstract convex analysis.

There are a number of algorithms of global minimization, based on abstract convexity. These methods use abstract convex analysis (see [17, 21, 25–27, 30]) in order to approximate the objective function by generalized affine minorants. Usually, they are extensions of the cutting plane method for convex optimization [16], as, for example, the cutting angle method for minimizing increasing convex-along-rays functions (see [1, 3, 4, 28, 29]). These methods proved efficient for problems of moderate size. However, they include the solution of a subproblem of minimizing the maximum of min-type functions, which is NP-hard. This paper is devoted to the algorithms for solving this subproblem.

There are several possible approaches. One of them is direct decomposition, which is applicable only to problems with a small number of variables. The second one is the reduction to a problem of mixed integer programming, which is inconvenient, as the number of binary variables grows fast after each iteration. So, we have chosen an approach, which is based on disjunctive programming (see [7–9]). This approach is convenient, as the subproblem with min-type functions can be easily reduced to a problem of disjunctive programming by introducing an auxiliary variable. Thus, we solve a problem of disjunctive programming instead of the original subproblem. Note that the problem with min-type functions, which arises in the minimization of increasing positively homogeneous functions, can also be solved efficiently by a different technique (see [5]).

Our algorithm uses relaxation and arbitration as its main parts. The disjunction with the maximum infeasibility or with the largest corresponding slack is chosen for arbitration and the term of the disjunction with the smallest infeasibility replaces the surrogate of the disjunction.

Note that in order to solve problems of global optimization of Lipschitz functions, we apply the cutting angle method (see [25, 27]). The initial problem is reduced to minimizing an increasing convex-along-rays function (see [28]). Thus we solve a sequence of subproblems with the objective functions, which are non-homogeneous (min-type functions plus a constant).

There are many possible applications. One of them is to solve the Hamiltonian cycle problem via min-type functions and Markov chains (see [11, 12]). This leads to a problem of disjunctive programming with the known optimal value, equal to zero. Another possible application is to use the approach for solving mathematical problems with equilibrium constraints (see [22]), which can be also represented with min-type functions.

2. Problem with min-type functions

Let p, q be vectors from $\mathbb{R}^n_{++} = \{x \in \mathbb{R}^n : x_i > 0 \ \forall i = 1, 2, \ldots, n\}$. We denote

$$\langle p, q \rangle = \min_i \ p_i q_i \, .$$

We address the following problem:

$$\begin{aligned} &\min \ \max_{i=1,2,\ldots,k} \{\langle l_i, x \rangle + b_i\} \\ &\text{s.t.} \quad Ax \leqslant d \, , \\ &\qquad 0 \leqslant x \leqslant m \, , \end{aligned} \tag{1}$$

where A is a matrix $m \times n$ and $l_i \in \mathbb{R}^n_{++}$, $b \in \mathbb{R}^k$, $m \in \mathbb{R}^n$, $d \in \mathbb{R}^m$. This problem is very important for minimizing Lipschitz functions, increasing convex-along-rays functions and increasing star-shaped functions (see [26–28]). One of the main of its applications is the cutting angle method for minimizing increasing convex-along-rays functions and Lipschitz functions (see [3, 4, 27]).

We recall the scheme of the algorithm for minimizing a Lipschitz function. Let us consider a Lipschitz programming problem in the following form:

$$\begin{aligned} &\min \left(M + f\left(\frac{y}{\sum\limits_{i=1}^{n} y_i} \right) \right) \cdot \left(\sum_{i=1}^{n} y_i \right)^p \\ &\text{s.t.} \quad g(y) \leqslant 0 \, , \\ &\qquad \sum_{i=1}^{n} y_i = 1 \, , \\ &\qquad y > 0 \, . \end{aligned} \tag{2}$$

Here f is the objective Lipschitz function, g is a Lipschitz function of constraints. Suppose that the parameters M and p are chosen in such a way that the objective function of problem (2) is increasing convex-along-rays (see [27]). This means that the function is increasing on $\mathbb{R}^n_+$ and its restriction on each ray in $\mathbb{R}^n_+$ is a convex function. The scheme of the algorithm is the following.

2.1. CUTTING ANGLE METHOD FOR LIPSCHITZ FUNCTIONS

Step 0. Let $k := 0$. Choose an arbitrary point $y_0 \in S_n = \{x \in \mathbb{R}^n_+ : \Sigma^n_{i=1} x_i = 1\}$ such that $g(y_0) \leqslant 0$ and parameters $M > 0,\ p > 0$.

Step 1. Calculate a vector ℓ_k with the coordinates ℓ_{ki}:

$$\ell_{ki} = \frac{p(f(y_k) + M)}{y_{ki}}, \quad \text{if } y_{ki} \neq 0\,; \quad \ell_{ki} = 0 \quad \text{if } y_{ki} = 0\,,$$

where y_{ki} is the ith coordinate of the current vector y_k.

Step 2. Define the function h_k:

$$h_k(y) = \min_{i,\, \ell_{ki} > 0} \ell_{ki} y_{ki} + (1 - p)(f(y) + M)\,.$$

Step 3. Find a global optimum of the problem:

$$\begin{aligned} &\min \max_{0 \leqslant i \leqslant k} h_i(y) \to \\ &\text{subject to} \quad y \in S\,, \\ &\qquad\qquad\ \ g(y) \leqslant 0\,. \end{aligned}$$

Let y^* be a solution of this problem.

Step 4. Let $k := k + 1$, $y_k := y^*$ and go to Step 1. Note that in Step 3 we should essentially solve the problem (1).

We can rewrite problem (1) as:

$$\begin{aligned} &\min t \\ &\langle l_i, x\rangle + b_i \leqslant t \quad i = \overline{1, k} \\ &\text{s.t.} \quad Ax \leqslant d \\ &\qquad 0 \leqslant x \leqslant m\,. \end{aligned} \tag{3}$$

There are two main approaches to solve the problem (3): directly or by applying a dychotomy with respect to t, thus obtaining a sequence of systems:

$$\begin{aligned} &\langle l_i, x\rangle \leqslant c_i \quad i = \overline{1, k} \\ &Ax \leqslant d \\ &0 \leqslant x \leqslant m\,, \end{aligned} \tag{4}$$

where $c_i = \bar{t} - b_i$ and $\bar{t}$ is fixed. Let $N = \{1, 2, \ldots, N\}$. Thus N denotes both the largest element of $\{1, 2, \ldots, \mathrm{N}\}$ and the set itself. Let $\vee$ mean the logical OR. The problem (3) can now be represented as:

$$\begin{aligned} &\max(-t) \\ &\text{s.t.} \quad \vee_{j\in N}(l_{ij}x_j - t) \leqslant -b_i \quad i=\overline{1,k}\,, \\ &\qquad Ax \leqslant d\,, \\ &\qquad 0 \leqslant x \leqslant m \end{aligned} \tag{5}$$

and the system (4) as:

$$\begin{aligned} &\vee_{j\in N} l_{ij}x_j \leqslant c_i \quad i=\overline{1,k} \\ &Ax \leqslant d \\ &0 \leqslant x \leqslant m\,. \end{aligned} \tag{6}$$

Each matrix in the disjunction i in the system (3) has the following form:

$$\begin{bmatrix} l_{i1} & 0 & .. & 0 & -1 \\ 0 & l_{i2} & \dots & 0 & -1 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & l_{in} & -1 \end{bmatrix}.$$

In the second case each matrix in the disjunction is just the diagonal matrix:

$$\begin{bmatrix} l_{i1} & 0 & \dots & 0 \\ 0 & l_{i2} & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & l_{in} \end{bmatrix}.$$

The typical approach to solving (4) is via its reduction to a mixed integer programming problem. We obtain the following system (j changes from 1 to k) of constraints:

$$\begin{aligned} &l_{j1}x_1 - My_{j1} \leqslant b_j \\ &l_j x_2 - My_{j2} \leqslant b_j \\ &\dots\dots\dots\dots\dots \\ &l_{jn}x_n - My_{jn} \leqslant b_j \\ &\sum_{i=1}^{n} y_{ji} \leqslant n-1\,, \\ &Ax \leqslant d\,, \\ &0 \leqslant x \leqslant m\,, \end{aligned}$$

where $y_{ji} \in \{0, 1\}$ for all i, j, and M is a large positive constant.

Note that the solution of the problem (3) is crucial for global optimization of wide classes of functions, but for the moment there have been no efficient algorithm developed for solving (3). Direct decomposition is applicable only for small directions, and the reduction to mixed integer linear programming leads to the growth of the dimensionality of the problem by n binary variables at each iteration of the cutting angle method (see [3, 4]). Numerical experiments have shown that this reduction is usually inefficient for sufficiently large dimensionality of the problem.

We propose to apply to the solution of (3) methods of disjunctive programming (see [7, 9]), as (5) and (6) are disjunctive programming problems.

3. An algorithm for disjunctive programming problems

In this section we follow F. Beaumont (see [9]), who developed a method of solving disjunctive programming problems without their reduction to mixed integer programming problems. The method proved its high numerical efficiency for various problems, outperforming conventional codes.

Let us consider the following problem:

$$\begin{aligned}
&\max \sum_{j\in N} c_j x_j \\
&\text{subject to} \\
&\sum_{j\in N} a_{ij}x_j \leqslant b_i\,, \quad i\in M\,, \\
&0\leqslant x_j \leqslant m_j\,, \quad j\in N\,, \\
&\vee_{i\in D_l} \sum_{j\in N} a_{ij}x_j \leqslant b_i\,, \quad l\in L\,.
\end{aligned} \tag{7}$$

Denote $u_i = \Sigma_{j\in N} m_j a_{ij}^+ - b_i$ for all i, where $a_{ij}^+ = \max\{a_{ij}, 0\}$.

A relaxation of the problem (7), obtained by replacing each disjunction by its surrogate, is the following linear programming problem:

$$\begin{aligned}
&\max \sum_{j\in N} c_j x_j \\
&\text{s.t.} \quad \sum_{j\in N} a_{ij}x_j \leqslant b_i\,, \quad i\in M \\
&\qquad 0\leqslant x_j \leqslant m_j\,, \quad j\in N \\
&\qquad \sum_{i\in D_l} \frac{1}{u_i} \sum_{j\in N} a_{ij}x_j \leqslant d_l - 1 + \sum_{i\in D_l} \frac{b_i}{u_i}\,, \quad l\in L\,.
\end{aligned} \tag{8}$$

Here $d_l = |D_l|$ is the number of elements in D_l. The last constraint is called the surrogate for the disjunction, corresponding to the index set D_l.

The dual problem to (8) is the following:

$$\begin{aligned}
&\min \sum_{i\in M} b_i Y_i + \sum_{j\in N} m_j Z_j + \sum_{l\in L} \left(d_l - 1 + \sum_{i\in D_l} \frac{b_i}{u_i}\right) Q_l \\
&\text{s.t.} \quad \sum_{i\in M} a_{ij}Y_i + Z_j + \sum_{i\in D_l} \left(\frac{Q_l}{u_i}\right) a_{ij} \geqslant c_j\,, \quad j\in N,\ l\in L\,.
\end{aligned} \tag{9}$$

Here Q_l are shadow prices for the disjunction with the index set D_l.

Instead of using last constraints in (8) as surrogates, it is possible to apply a different approach, which is based on a theorem due to Balas.

THEOREM 3.1. [7] (*Balas*, 1979) *The constraint* $\alpha x \leqslant \beta$ *is a surrogate of the system*

$$\vee_{i \in D}(A^i x \leqslant b^i) \quad x \geqslant 0$$

if and only if there exists $\Psi^i \geqslant 0$, $i \in D^*$ *such that*

$$\alpha \leqslant \Psi^i A^i \quad i \in D^*,$$
$$\beta \geqslant \Psi^i b^i \quad i \in D^*,$$

where A^i *is* $m \times n$ *matrix,* Ψ^i *is a vector* $1 \times m_i$, $b \in \mathbb{R}^m$ *and* D^* *is a subset of* D *for which the parts* $i \in D$ *of the disjunction are feasible.*

We suppose that each matrix A^i has n common terms, given by the upper bounds for the variables m_j, and one general constraint. Each member of the disjunction has then the following form:

$$\begin{bmatrix} x_j \leqslant m_j, & j \in N \\ \Sigma_{j \in N}\, a_{ij} x_j \leqslant b_i, \end{bmatrix} \tag{10}$$

where $i \in D$.

Here the square bracket means that at least one of the constraints must be satisfied.

We compute the value

$$\phi_i = \frac{1}{\sum\limits_{j \in N} a_{ij} m_j - b_i}$$

If the denominator is negative, we take instead of it one of the following values:

$$\phi_i = \frac{1}{\sum\limits_{j \in N} a_{ij}^+ m_j - b_i}$$

or

$$\phi_i = \frac{1}{\sum\limits_{j \in N} a_{ij} m_j - b_i}.$$

As noted in [9], the first choice is better for computational performance.

Then, let

$$\alpha_j = \max_{i \in D}(a_{ij} \phi_i) \quad j \in N.$$

Also let

$$\theta_j^i = \alpha_j - a_{ij}\phi_i\,, \quad j \in N,\ i \in D\,,$$

$$\beta = \max_{i \in D}\left(\sum_{j \in N} m_j \theta_j^i + \phi_i b_i\right).$$

The surrogate constraint is thus

$$\alpha x \leqslant \beta\,. \tag{11}$$

We shall have l such constraints initially.

The algorithm for solving disjunctive programming problems is the following [9].

3.1. ALGORITHM FOR SOLVING DISJUNCTIVE PROBLEMS

Step 0. Generate a relaxation of the initial problem, replacing each disjunction by its surrogate (11). The set U of unsolved problems includes initially only this problem. Set BEST $= -\infty$. Do Steps 1, 2, 3 until U becomes empty.

Step 1. Delete a chosen problem P from U and solve it as a linear programming problem. Denote its optimal value z ($z = -\infty$, if the problem is infeasible).

Step 2. If $z >$ BEST and we obtained a disjunctive solution, let BEST $= z$ and store this solution.

Step 3. If $z >$ BEST, but we have not obtained a disjunctive solution, choose a disjunction and add to U the problems, obtained by replacing the disjunction by its terms.

Step 4. An optimal solution is the last one, remembered in Step 2.

There are two main problems to be resolved in this algorithm. The first one is the choice of the disjunction in Step 3 to arbitrate. It has been proposed to evaluate the infeasibility of each disjunction as:

$$\delta = \min_{i \in D_l}\left\{\sum_{j \in N} a_{ij}x_j - b_i\right\}.$$

Then the disjunction with the largest infeasibility δ should be arbitrated.

Other approaches are to choose a disjunction with the maximum absolute value of the shadow price Q_l, or with the smallest slack in the surrogate constraint. In any case, the term with the smallest infeasibility $\Sigma_{j \in N}\, a_{ij}x - b_i$ should be chosen for replacing the surrogate.

4. Solving the problem with min-type functions

We think that some modifications of the algorithm, described above, can be useful. This is important, because in [9] the way of choosing the problem from the list of unsolved problems was not indicated. This choice is crucial, however, for the numerical efficiency of the algorithm. The benefits of modifications of the method (which are the main new result of this paper) have been confirmed by numerical experiments.

Let Ω be the set of solved linear programs (in Step 1). Each problem is defined by the vector:

$$P_j = (Y_1, Y_2, \ldots, Y_l) \quad j \in J_k ,$$

where k is the number of iterations. There are two possibilities for each Y_i.
(1) $Y_i = S$, which means that the surrogate is used instead of the disjunction i.
(2) $Y_i = m$, which means that the term m of the disjunction i is used instead of it.

DEFINITION 4.1. The set of the problems is called the full set, if replacing all the letters S in each problem by all possible numbers of terms of the disjunctions, we obtain all possible problems, obtained by replacing some (one, or maybe, all) disjunctions by its terms.

We shall maintain full sets of linear programming problems.

Each problem j is characterized by the following properties and values:
(1) Feasible/infeasible.
(2) The optimal value z_j.
(3) The slacks of surrogate constraints.
(4) The slacks of constraints, which are terms of disjunctions.
(5) The shadow prices.

We assume that no optimal disjunctive solution is known, the set of solved problems is full, and the question is, which new problem to choose. Once a problem has been chosen, we use standard criteria for the choice of the disjunction to arbitrate and its term, unless the problem has been solved.

The first possibility is to choose first a solved problem with the largest z_j. Then, we choose a disjunction with the largest infeasibility and the term in it with the smallest infeasibility. If the problem thus obtained is in Ω, we take the term in the same disjunction with the second smallest infeasibility, and so on. If all the terms have been considered, and all corresponding problems are in Ω, we take the disjunction with second largest infeasibility and consider its terms, and so on.

It is desirable, however, to limit the number of steps, in which we deal with the problem with the largest z_j. If after solving $K > 0$ problems, where K is a positive parameter, we have not obtained a disjunctive solution with $z > BEST$, we take a solved problem with the second best optimal value. Obviously, we should not repeat the same problem twice.

There is a possibility that all problems in Ω are infeasible. Then we suggest to take for arbitration a problem with the smallest maximum slack of the constraints.

The second possibility is to choose first a solved problem with the smallest number of surrogate constraints. If for two problems such numbers are equal, the problem with the larger z_j is being chosen. This sorts all the problems from Ω in a certain order. There are several cases:

(1) We obtained a problem in which $Y_i \neq S$ for all i. We store its optimal solution and do not deal with it any more.
(2) After $M > 0$ iterations, at each of which the number of surrogate constraints is reduced, we have not obtained a disjunctive solution with $z > BEST$. Then we take the next problem from the list, corresponding to our ordering. Note that for each problem in Ω we must remember its children, that is, problems, obtained by replacing one or more surrogate constraints in it by terms of disjunctions. If our criterion of choosing a disjunction or its term gives us a child, which is already in Ω, we take the second best disjunction (term) with respect to our criterion and so on.
(3) After some number of arbitrations we discover that the corresponding linear programming problem is infeasible. Then we do the same, as in the second case. However, if all problems in Ω are infeasible, we take the problem with the smallest of the largest slacks of the constraints.

Of course, the above two approaches for the choice of the problem can be combined. Note that the set Ω changes after each iteration (one solved problem is added to it).

Now, let us see which surrogate constraints we shall obtain for problems (3) and (4), using the two approaches, described above. Let m_{n+1} be the upper bound for t (assume that it is known).

In the first approach (see problem (8)), substituting the appropriate matrices, for problem (3) we obtain the surrogate constraints:

$$\sum_{i \in D_k} \frac{1}{u_i} (l_{ki} x_i - t) \leqslant n - 1 - \sum_{i \in D_k} \frac{b_i}{u_i}, \quad k \in L,$$

where $D_k = |N|$ and $u_i = m_i l_{ki} + b_i + m_{n+1}$.

For problem (4) we obtain the following surrogate constraints:

$$\sum_{i \in D_k} \frac{1}{u_i} l_{ki} x_i \leqslant n - 1 - \sum_{i \in D_k} \frac{b_i}{u_i}, \quad k \in L,$$

where $D_k = |N|$ and $u_i = m_i l_{ki} + b_i + m_{n+1}$, $i \in D_k$.

If we use the Balas theorem, the surrogate constraints will be different. Each ith term of the disjunction number k will be the following system:

$$x_j \leqslant m_j \quad j \in N,$$
$$t \leqslant m_{n+1},$$
$$l_{ki} x_i - t \leqslant -b_k.$$

The corresponding matrix is

$$A_{ki} = \begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & 1 \\ 0 & \dots & l_{ki} & \dots & -1 \end{bmatrix}$$

Then

$$\phi_{ki} = \frac{1}{l_k i m_i - m_{n+1} b_k},$$
$$\alpha_{kj} = l_{kj}\phi_{kj} \quad j \in N,$$
$$\alpha_{k,n+1} = \max_{i \in D}(-\phi_i)$$

As

$$\theta^i_{kj} = l_{kj}\phi_{kj} - a_{kij}\phi_{ki} \quad j \in N,$$

we have

$$\theta^j_{kj} = 0; \quad j = \overline{1, n} \qquad \theta^i_{kj} = l_{kj}\phi_{kj}, \quad i \neq j,\ j \in N,$$
$$\theta^i_{k,n+1} = \max_{i \in D}(-\phi_{ki}) + \phi_{ki}.$$

Knowing θ^i_j, it is easy to compute the right parts of the surrogates $\alpha_k x \leqslant \beta_k$ (see above).

The dual problem in the case of the problem (3) is the following:

$$\min \sum_{k \in L} \left(n - 1 - \sum_{i \in D_k} \frac{b_k}{u_i} \right) Q_k + \sum_{i \in M} d_i Y_i + \sum_{i=1}^{n+1} m_i Z_i$$

subject to

$$\sum_{i \in M} a_{ij} Y_j + Z_j + \frac{Q_k}{u_j} l_{kj} \geqslant 0 \quad j \in N,\ k \in L$$

$$\sum_{i \in M} a_{ij} Y_j + Z_{n+1} - \sum_{i \in D_k} \frac{Q_k}{u_i} \geqslant -1 \quad k \in L.$$

Here $u_i = m_i l_{ki} + m_{n+1} + b_i$. The values Q_k. as above, are the shadow prices, and it can be advantageous to choose the disjunction with the maximum absolute shadow price for arbitration.

5. Applications

There are many possible applications of our approach. We consider only two of them. The first one is solving the Hamilton Cycle Problem (HCP).

5.1. FORMULATION

We consider the Hamilton cycle problem (HCP) from a somewhat unorthodox perspective of an embedding in a Markov Decision Process (MDP) that was developed in [11] and [12].

More precisely, the form of the HCP analysed here is the following: Given a directed graph G with N nodes find a simple cycle of N arcs, that is a Hamiltonian cycle, or determine that none exist.

The unorthodox formulation of the HCP that forms the basis of the present investigation is as the problem of finding a global minimum (with the objective function value equal to 0) of a suitably constructed indefinite quadratic program:

$$\begin{aligned} \min \quad & x'Vx \\ \text{s.t.} \quad & Ax = b\,, \\ & x \geqslant 0\,. \end{aligned} \tag{12}$$

Notation. Assume that G has no self-loops and let $A(i)$ be the set of arcs emanating from node i. Assume that

$$n_i = |A(i)| \geqslant 1 \quad \forall i \in E = \{1, 2, \ldots, N\}\,.$$

Thus there is at least one arc emanating from each node for otherwise the HCP would be trivial. An arc emanating from node i will be an ordered pair (i, a).

Whenever it is obvious what the 'tail' of an arc is, we will denote the arc only by its 'head'. For instance, if we are at node i and we are considering an arc (i, a) we will denote it merely by a. Let a perturbation parameter $\varepsilon \in (0, 1)$ be fixed and define

$$d_N(\varepsilon) = 1 + \sum_{k=2}^{N-2} (1 - \varepsilon)^{k-2}\,.$$

Furthermore, for each $i,\ j \in E$ and $a \in A(i)$, as in [11], define the coefficients $p_{ija}(\varepsilon)$ by:

$$P_{ija}(\varepsilon) = \begin{cases} 1 & \text{if } i = 1 \text{ and } a = j \\ 0 & \text{if } i = 1 \text{ and } a \neq j \\ 1 & \text{if } i > 1 \text{ and } a = j = 1 \\ \varepsilon & \text{if } i > 1,\ a \neq j, \text{ and } j = 1 \\ 1 - \varepsilon & \text{if } i > 1,\ a = j, \text{ and } j > 1 \\ 0 & \text{if } i > 1,\ a \neq j, \text{ and } j > 1\,. \end{cases}$$

Now, matrix A and vector b of (12) are defined by the following system of linear constraints:

$$(C1) \quad \sum_{i \in E} \sum_{a \in A(i)} (\delta_{ij} - p_{ija}(\varepsilon)) x_{ia} = 0\,, \quad j \in E$$

$$(C2) \quad \sum_{i \in E} \sum_{a \in A(i)} x_{ia} = 1$$

$$(C3) \quad \sum_{a \in A(i)} x_{ia} = 1/d_N(\varepsilon)$$

$$(C4) \quad x_{ia} \geqslant 0; \quad i \in E,\ a \in A(i)\ .$$

Here δ_{ij} is the Kronecker delta. If x satisfies $(C1)$–$(C4)$ and $x'Vx = 0$, then the positive entries of x identify a Hamiltonian cycle of G.

Thus A is an $(N+2) \times (\Sigma_{i=1}^{N} n_i)$ matrix and $b' = (0, 0, \ldots, 1, 1/d_N(\varepsilon))$ is an $(N+2)$-dimensional vector. Of course, δ_{ij} is the Kronecker delta. The matrix V is an $(\Sigma_{i=1}^{N} n_i) \times (\Sigma_{i=1}^{N} n_i)$ block diagonal matrix whose ith diagonal block is an $n_i \times n_i$ matrix $V_i = J_i - I_i$ where J_i is a matrix with 1 in every entry and I_i an identity matrix of an appropriate dimension.

Of course, the global optimization of (12) is, in rinciple, very difficult problem since V is an indefinite matrix. In our case we note (see also [2]) that for x feasible for (12)

$$x'Vx = 0 \tag{13}$$

if and only if

$$\sum_{i=1}^{N} \sum_{a \neq b} x_{ia} x_{ib} = 0\ . \tag{14}$$

which, in turn holds if and only if

$$\sum_{i=1}^{N} \min\{x_{ia}, x_{ib}\} = 0\ . \tag{15}$$

Thus the problem of determining whether the global minimum of (12) has the objective function value equal to 0 at some feasible point can be answered by solving the 'Min-type' global optimization problem (see [2]):

$$\min_{x} \max_{i} \min_{a \neq b} \min\{x_{ia}, x_{ib}\} \tag{16}$$

subject to: x satisfies $(C1)$–$(C4)$.

As the optimal value of the problem is zero and the variables are positive, we can rewrite (16) as:

$$(CD) \quad x_{ia} \vee x_{ib} = 0 \quad \forall a \neq b, \quad \forall i\ .$$

Then we can solve the roblem with the constraints $(C1)$–$(C4)$ and (CD) by the disjunctive approach, outlined in previous section.

In many cases the formulation of the problem with min-type functions is preferable to the indefinite quadratic programming because some linearity properties are still preserved. The nonsmoothness of the objective function is not a disadvantage, since the derivatives are useful for finding a local minimum, and for solving HCP a global one is needed.

The second possible application is to linear problems with equilibrium constraints [22]. These are typically formulated as follows:

$$Ax \leqslant b\,,$$

$$Cy \leqslant d\,,$$

$$[x, y] = 0\,,$$

$$x, y \geqslant 0\,,$$

where the square brackets mean the inner product. Here A, C are $m \times n$ matrices and b, $d \in \mathbb{R}^n$.

We can reformulate the problem in the following way:

$$x_i \vee y_i = 0 \quad \forall i$$

$$Ax \leqslant b\,,$$

$$Cy \leqslant d\,.$$

This is a problem, for which we can apply the methods presented above.

Another possible application is in the sublinear algebra problems, which are important for the quasi-differential calculus (see [10]). Typically, we have a set of the following constraints:

$$\min_{i=\overline{1,m}} l_i x_i + \max_{j=\overline{1,k}} r_j x_j = b\,.$$

Each constraint can be represented in a disjunctive form:

$$l_1 x_1 + r_1 x_1 \qquad = b$$

$$l_1 x_1 \leqslant l_i x_i \qquad i \neq 1$$

$$r_1 x_1 \geqslant r_j x_j \qquad j \neq 1$$

OR

$$l_1 x_1 + r_2 x_2 \qquad = b$$

$$l_1 x_1 \leqslant l_i x_i \qquad i \neq 1$$

$$r_2 x_2 \geqslant r_j x_j \qquad j \neq 2$$

OR

$$\ldots\ldots\ldots\ldots\ldots$$

$$l_m x_m + r_k x_k \qquad = b$$

$$l_m x_m \leqslant l_i x_i \qquad i \neq m$$

$$r_k x_k \geqslant r_j x_j \qquad j \neq k$$

Obviously, the Balas theorem can be applied in order to generate surrogate constraints. It is necessary to generalize the algorithm, however, in order to solve

such problem. Certainly, each equality constraint can be represented as two inequalities.

6. Numerical experiments

A number of test problems were solved by the cutting angle method (see [25, 27]). The method for minimizing the maximum of min-type functions was the algorithm proposed in this paper. In order to improve the performance, the local search was used for improving the current iterate. For the local search either the bundle Newton method (see [19]) or Nelder-Mead simplex algorithm (see [20]) was used.

Let us consider first the following test examples. In Examples 1–2 the feasible set is the unit simplex

$$S_n = \left\{ x \in \mathbb{R}^n_+ : \sum_{j=1}^{n} x_j = 1 \right\}.$$

EXAMPLE 1.

$$\min_x \max_{i=1,\dots,10} \min_{j=1,\dots,5, l^i_j > 0} l^i_j x_j ,$$

where l^i, $i = 1, \dots, 10$ are rows of the following matrix:

$$L = \begin{pmatrix} 10.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 10.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 10.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 10.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 10.0 \\ 50.0 & 50.0 & 50.0 & 50.0 & 50.0 \\ 210.0 & 42.0 & 42.0 & 42.0 & 42.0 \\ 41.52488 & 174.40449 & 41.52488 & 41.52488 & 41.52488 \\ 176.09952 & 42.40819 & 42.40819 & 42.40819 & 42.40819 \\ 41.50719 & 176.02451 & 41.50719 & 41.50719 & 41.50719 \end{pmatrix}$$

The optimal value is equal to 2.38095. The best found value by the cutting angle method is 2.38095 and the absolute precision is 0.

EXAMPLE 2.

$$\min_x \max_{i=1,\dots,30} \min_{j=1,\dots,10, l^i_j > 0} l^i_j x_j ,$$

where l^i, $i = 1, \dots, 30$ are rows of the following matrix:

$$L = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 17 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 9 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 14 \\
2 & 11 & 2 & 31 & 45 & 1 & 5 & 9 & 11 & 1 \\
21 & 1 & 4 & 3 & 5 & 7 & 9 & 12 & 6 & 5 \\
7 & 7 & 4 & 29 & 19 & 7 & 3 & 4 & 11 & 8 \\
8 & 3 & 31 & 2 & 13 & 51 & 1 & 5 & 9 & 2 \\
22 & 7 & 17 & 18 & 1 & 12 & 6 & 6 & 7 & 13 \\
11 & 8 & 5 & 2 & 22 & 31 & 1 & 23 & 5 & 16 \\
14 & 13 & 1 & 31 & 19 & 18 & 3 & 5 & 4 & 28 \\
17 & 34 & 15 & 40 & 1 & 7 & 4 & 7 & 13 & 51 \\
23 & 3 & 5 & 17 & 27 & 3 & 6 & 1 & 15 & 3 \\
42 & 7 & 4 & 22 & 13 & 21 & 2 & 15 & 4 & 2 \\
17 & 4 & 14 & 6 & 17 & 21 & 19 & 3 & 19 & 15 \\
21 & 24 & 5 & 7 & 18 & 42 & 1 & 23 & 9 & 17 \\
13 & 22 & 2 & 37 & 8 & 9 & 11 & 1 & 8 & 11 \\
5 & 11 & 12 & 7 & 3 & 2 & 24 & 45 & 6 & 3 \\
8 & 21 & 32 & 4 & 9 & 1 & 23 & 52 & 2 & 1 \\
19 & 31 & 2 & 35 & 4 & 9 & 12 & 3 & 31 & 9 \\
18 & 2 & 23 & 41 & 9 & 6 & 16 & 6 & 26 & 4 \\
7 & 4 & 12 & 7 & 31 & 5 & 9 & 1 & 14 & 6 \\
29 & 7 & 12 & 23 & 7 & 1 & 13 & 9 & 4 & 2 \\
41 & 1 & 4 & 14 & 5 & 9 & 23 & 12 & 3 & 12
\end{pmatrix}$$

Approximate solution:

$$x^* = (0.269,\ 0.067,\ 0.090,\ 0.016,\ 0.134,\ 0.030,\ 0.038,\ 0.269,\ 0.067,\ 0.019)\,,$$

$$f(x^*) = 0.269\,.$$

The best found value by the cutting angle method is 0.272 and the absolute precision is 0.003. Similar examples were solved by this method for five to 10 variables with the same precision.

Let us consider examples of larger dimensionality. In these examples Lipschitz functions are minimized by the technique from [25, 27]. The number of variables was equal to $n = 15$.

EXAMPLE 3 [13, 6].

$$f(x) = \frac{1}{d}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1\,,$$

$$\sum_{i=1}^{n} x_i \leqslant 400\,, \quad x_i \geqslant -50,\ i = 1, \ldots, n\,,$$

$$d = 4000\,, \qquad x^* = (0, 0, \ldots, 0)\,, \qquad f(x^*) = 0\,.$$

Here x^* is the global optimum. The precision by norm obtained was 10^{-5}.

EXAMPLE 4 [18, 6].

$$f(x) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2(1 + 10\sin^2(\pi y_i + 1)) + (y_n - 1)^2(1 + \sin^2(2\pi x_n))\,,$$

$$y_i = 1 + 0.25 \times (x_i - 1)\,, \qquad \sum_{i=1}^{n} x_i \leqslant 70\,, \quad x_i \geqslant -1,\ i = 1, \ldots, n\,,$$

$$x^* = (1, 1, \ldots, 1)\,, \qquad f(x^*) = 0\,.$$

The precision by norm obtained was 10^{-6}.

In all cases the time of computation varied between 0.1 and 30.0 seconds. The software has been written in Microsoft Fortran 90 for Windows 98 on IBM Pentium MMX 400. The packages PNEW by L. Luksan (see [19]) and COBYLA2 by M.J.D. Powell (see [24]) were used for nonlinear optimization. The package LPDUAL was used for solving auxiliary linear programming programs. In all cases the cutting angle method outperformed by time and precision the global optimization code, based on simulated annealing. A number of problems with 15–20 variables were solved. The typical time of computation was 20–30 seconds.

Acknowledgments

This research has been supported by the Australian Research Council Grant A69701407 and RFFI grant 99-01-00173. The author is very grateful to A. Rubinov and A. Bagirov for their advice and help and to J. Filar and an anonymous referee for their useful remarks, which improved the quality of this paper.

References

[1] Abasov, T.M. and Rubinov, A.M. (1994), On the Class of H-Convex Functions, *Russian Acad. Sci. Dokl. Math.* 48, 95–97.

[2] Andramonov, M., Filar, J., Pardalos, P. and Rubinov, A.M. (2000), Hamiltonian Cycle Problem via Markov Chains and Min-Type Approaches, in Pardalos, P.M. (ed.), Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems, Kluwer Academic Publishers, pp. 31–47.

[3] Andramonov, M.Yu., Rubinov, A.M. and Glover, B.M. (1999), Cutting Angle Methods in Global Optimization, *Applied Mathematics Letters*.

[4] Andramonov, M.Yu., Rubinov, A.M. and Glover, B.M. (1997), Cutting Angle Method for Minimizing Increasing Convex-along-rays Functions, Research Report, University of Ballarat.
[5] Bagirov, A.M. and Rubinov, A.M., Global Minimization of Increasing Positively Homogeneous Functions over the Unit Simplex, Research Report, University of Ballarat.
[6] Bagirov, A.M. and Rubinov, A.M. (2001), Cutting Angle Method and a Local Search, Research Report, University of Ballarat, 2001.
[7] Balas, E. (1979), Disjunctive Programming, *Annals of Discrete Mathematics* 5, 3–51.
[8] Balas, E., Tama, J. and Tind, J. (1989), Sequential Convexification in Reverse Convex and Disjunctive Programming, *Mathematical Programming* 44(3), 337–350.
[9] Beaumont, F. (1990), Algorithm for Disjunctive Programming Problems, *European Journal of Operational Research*.
[10] Demyanov, V.F. and Rubinov, A.M. (1995), *Constructive Non-smooth Analysis*, Frankfurt, Peter Lang.
[11] Filar, J.A. and Krass, D. (1995), Hamiltonian Cycles and Markov Chains, *Mathematics of Operations Research* 19(1), 223–237.
[12] Chen, M. and Filar, J.A. (1992), Hamiltonian Cycles, Quadratic Programming, and Ranking of Extreme Points, Floudas, P. and Pardalos, P. (eds.), *Global Optimization*, Princeton University Press, Princeton, NJ.
[13] Griewank, A. (1981), Generalized Descent for Global Optimization, *Journal of Optimization Theory and Applications* 34, 11–39.
[14] Horst, R. and Pardalos, P. (eds.) (1996), *Handbook on Global Optimization*, Kluwer Academic Publishers, Dordrecht.
[15] Horst, R. and Tuy, H. (1990), *Global Optimization*, Springer, Berlin.
[16] Kelley, J. (1960), The Cutting Plane Method for Solving Convex Programs, *SIAM Journal* 8(4), 703–712.
[17] Kutateladze S.S. and Rubinov, A.M. (1976), *Minkowski Duality and its Applications*, Nauka, Novosibirsk.
[18] Levy, A.V. and Montalvo, A. (1985), The Tunnelling Algorithm for the Global Minimization of Functions, *SIAM Journal of Scientific and Statistical Computing* 6, 15–29.
[19] Luksan, L. and Vlcek, J. (2000), NDA: Algorithms for Nondifferentiable Optimization. Research Report V-797, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic.
[20] Nelder, J.A. and Mead, R. (1965), A Simplex Method for Function Minimization, *Computer Journal* 7, 308–313.
[21] Pallaschke, D. and Rolewicz, S. (1997), *Foundations of Mathematical Optimization*, Kluwer Academic Publishers, Dordrecht.
[22] Pang, J.S., Ralph, D. and Luo, S. (1997), *Mathematical Programs with Equilibrium Constraints*, Wiley, New York.
[23] Pinter, J. (1996), *Global Optimization in Action. Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications*, Kluwer Academic Publishers, Dordrecht.
[24] Powell, M.J.D., Direct Search Algorithms in Optimization Calculations, Technical Report DAMTP 1998/NA04.
[25] Rubinov, A.M. (2000), *Abstract Convexity and Global Optimization*, Kluwer Academic Publishers, Dordrecht.
[26] Rubinov, A.M. and Andramonov, M.Yu. (1999), Minimizing Increasing Star-Shaped Functions Based on Abstract Convexity, *Journal of Global Optimization*, August, 29–49.
[27] Rubinov, A.M. and Andramonov, M.Yu. (1999), Lipschitz Programming via Increasing Convex-along-rays Functions, *Optimization Methods and Software*.
[28] Rubinov, A.M. and Glover, B.M. (1996), Increasing Convex-along-Rays Functions with Applications to Global Optimization, Research Report 21/96, University of Ballarat.

[29] Rubinov, A.M. and Glover, B.M. (1999), Increasing Convex-along-Rays Functions with Applications to Global Optimization, *Journal of Optimization Theory and Applications* 102, 615–642.
[30] Singer, I. (1997) *Abstract Convex Analysis*, Wiley & Sons, New York.