Published in:
Computational Mathematics and Mathematical Physics, RF, 2003, v. 43, no. 7, pp. 951–960.

CONCAVE PROGRAMMING UNDER THE SIMPLEST LINEAR CONSTRAINTS

2003, A.I. RUSAKOV

We present a stable algorithm of concave programming on a feasible polyhedron, determined by finite bounds for the variables and a scalar condition of equality type. The algorithm is based on branch and bound scheme, where the inspected regions are cut off by successive tightening of the bounds.

1. Introduction

There are few basic schemes of concave programming under linear constraints, but the corresponding methods have a limited applicability on some reasons. The approximation-combinatorial method is most commonly used, but its applicability is determined by the type of the objective function and the size of the problem [1]. The conical methods are considered as very effective, and numerical results make appearance they are applicable for large-scale problems [2, 3]. But the author's attempt to use the conical method for optimizing on the feasible polyhedron of a high dimension (about 20) failed because of computational instability. In this paper we present the stable algorithm of the following maximization problem for the convex function $f(x)$, defined on E^m:

$$f(x) \longrightarrow \max; \tag{1}$$

$$\sum_{i=1}^{m} x_i = A; \tag{2}$$

$$\underline{x}_i \le x_i \le \bar{x}_i, \quad i = \overline{1,m}, \tag{3}$$

where $\underline{x}_i < \bar{x}_i$ for any i; feasible set M, defined by the relations (2) and (3), is not empty and not degenerated to the point.

It is additionally assumed about function $f(x)$, that the later is uniformly continuous on the set

$$U = \{x \colon x \in P \text{ and } f(x) \le f_{\max} + \Delta f_0\}, \tag{4}$$

where P is hyperplane (2), $f_{\max}$ is exact maximum of the function $f(x)$, Δf_0 is the given tolerence of the criterion function maximization.

The elaborated algorithm allows one to find an ε-optimal solution of this problem, i.e., a solution x^{opt} such that $f(x^{\text{opt}}) \ge f_{\max} - \varepsilon$, where $\varepsilon = \Delta f_0$.

The uniform continuity assumption is applicable for extensive class of the problems. For instance, a convex function on E^m is continuous on E^m [4]; therefore, (4) is a closed set. Suppose that this set is bounded (it's true for a broad class of functions $f(x)$); then it is compact set. A continuous function on a compact set is uniformly continuous [5]. Note, though, that uniform continuity has been used for the proof of an algorithm's finiteness, and is not obligatory in applications.

The problem (1)—(3) arises in performance control of high-accuracy systems, namely in testing the hypothesis about the variance of goal function, based on a series of regression experiments [6, 7].

Denote M^0 the hyperparallelepiped (HP) of the form (3). An HP, defined by the restrictions for any component (for example, (3)), we shall name correctly oriented. The denotations M^0, $\underline{x}_i$, $\bar{x}_i$, we shall use to refer not only to the initial HP defined with problem restrictions, but to any included into the initial and correctly oriented HP. Accordingly, we consider polyhedra in the hyperplane of the form

$$M = P \bigcap M^0.$$

The feasible set of problem (1)—(3) is one of such polyhedra. This extensions of terms are conditioned by the fact that, at each step of the proposed optimization algorithm, the bounds of HP M^0 are modified to specify the domain M that is not yet explored (not cut off).

2. The basic notations

We introduce a system of notations and conceptions to ground the solution of problem (1)—(3). The theorems are not proved, because the proofs are fairly simple and the analogous facts for polyhedra in the m-dimensional space are well known [8].

We define a polyhedron vertex in the hyperplane as the point x such that we have for some integer $I \geq 1$:

$$\begin{aligned} x_i^0 &= \underline{x}_i \text{ or } \bar{x}_i \text{ for } i \neq I; \\ x_I^0 &= A - \sum_{i \neq I} x_i^0. \end{aligned} \tag{5}$$

The vertex is said to be degenerated, if $x_I^0 = \underline{x}_I$ or $\bar{x}_I$, and non-degenerated, in opposite case. Note that, for degenerated vertex, the number I is not determined uniquely.

We define a polyhedron edge, adjacent to vertex x^0, as the m-dimensional segment of non-zero length, such that only two components vary over its extension:

$$\begin{aligned} x_i &= x_i^0 + \Delta; \\ x_I &= x_I^0 - \Delta, \end{aligned} \tag{6}$$

where $\Delta \geq 0$ if $x_i^0 = \underline{x}_i$ and $\Delta \leq 0$ if $x_i^0 = \bar{x}_i$, $i \neq I$; the values of Δ are defined with constraints (3). Each non-degenerated vertex has $m - 1$ adjacent edges. Each edge has two vertices as extreme points: one of them is the vertex appearing in the definition of the edge, another is named adjacent vertex.

We define a simplest cone in the hyperplane as a point set in E^n, specified by a system of linearly independent vectors B_{*i}, $i=\overline{1,m-1}$, and a cone vertex (point x_A), as follows:

$$x = \sum_{i=1}^{m-1} \lambda_i B_{*i} + x_A, \quad \lambda_i \geq 0. \tag{7}$$

A ray of the form $\lambda_i B_{*i} + x_A, \lambda_i \geq 0$, is said to be an edge of simplest cone.

<u>Theorem 1.</u> A polyhedron in the hyperplane is contained in the simplest cone, constructed from the non-degenerated vertex of the polyhedron and its adjacent edges.

We define a simplex as a portion of simplest cone specified with additional condition:

$$\sum_{i=1}^{m-1} \lambda_i < C, \quad C > 0. \tag{8}$$

m points x_A (the vertex of cone) and $x_A + CB_{*i}$, $i=\overline{1,m-1}$, is said to be vertices of simplex.

<u>Theorem 2.</u> The convex function reaches its maximum on the simplex at a vertex of this simplex.

The similar statement may easily be proved for a polyhedron M.

We define a cutting hyperplane as a hyperplane of the form

$$b^{\mathrm{T}}(x - x_A) = 1, \tag{9}$$

such, that the half-space

$$b^{\mathrm{T}}(x - x_A) \leq 1$$

cuts off simplex (8) from cone (7). Here the upper index “T” denotes transposition.

<u>Theorem 3.</u> For the simplex, given on a cone with a vertex x_A by the expressions (7) and (8) where $C = 1$, the cutting hyperplane (9) is determined by the vector b with the components

$$b_i = \sum_{k=1}^{m-1} B^{+}_{ki}, \tag{10}$$

where B^+ is a pseudoinverse of the matrix

$$B = (B_{*1}, \ldots, B_{*\,m-1}). \tag{11}$$

3. The solving algorithm

To simplify the description of the optimization algorithm we introduce the following assumption: for every polyhedron M created in the algorithm, the vertices being tested for the maximum by comparing of the objective function with a current highest value, are nondegenerate.

In practice this assumption is ensured in following way [9]: if degenerated vertex has been discovered, a small correction is introduced into the task's parameters (for instance, to correct the value A in the condition (2)).

We shall assume that one has to seek only the maximum of a function $f(x)$, i.e. no maximizer is required. For seeking a maximizer x^{opt}, every computation of the best (record) value $Rec = f(x)$ should be supplied with assignment: $x^{\text{opt}} = x$.

Presented optimization algorithm is based on the brunch-and-bound scheme, where every optimization step (brunching step) consist in operation of partitioning of the tested domain M into three subsets as follows:

1) a subset, that is tested for the maximum first, using internal cutting-off algorithm;
2) a subset, which analyzed with recursive run of brunching step;
3) a subset, that is tested in last turn by returning to the beginning of a current step. The probability of repeated brunching in this subset is not high.

Subsets 2 and 3 may be empty or not empty only simultaneously.

At the current step, one carries out the analysis and cutting of the first subset. Then one passes to the next step, which may be either analysis of the subset 2, if it's not empty, or analysis of the subset 3 of preceding step, in opposite case.

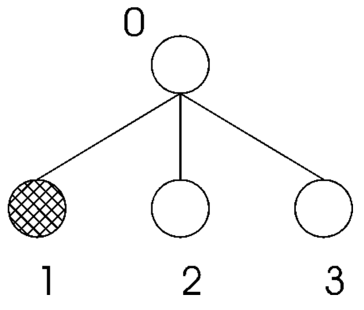

Fig.1

In Fig. 1 it is shown graphical representation of any optimization step. Vertex 0 depicts the original subset at the plot, vertices 1, 2 and 3 represent the above-mentioned subsets, numbered in a similar order. In Fig. 2 it is shown the sample of optimization process in the form of the rooted graph. In given case the root denotes the preparatory operations to the sequence of optimization steps. Hanging vertices of the graph are all the vertices of the first type and also the vertices of the second and third types, if they correspond to empty subsets (in Fig.2 such vertices are crossed over). Initial vertices of all steps are numbered in order to their succession.

Before passing to the next step of branching in the case of nonempty subset of the second type (i.e. before the recursion) one calculates the boundaries of HP M^0, which is smaller than at the previous step and determines the domain M to be investigated. Finally the size of HP M^0 becomes so small, that on the next step no subsets of second and third type are selected and the brunching terminates.

In the cutting algorithm the criterion value is determined at the one of vertices of the polyhedron M (at the point of the local maximum of $f(x)$). This value is assigned to variable Rec (the best value or the record of an algorithm), if it's greater then criterion value attained before. Then a neighborhood of considered vertex is constructed in M, so that the criterion is no greater within it than $Rec + \Delta f_0$. This neighborhood is cut off

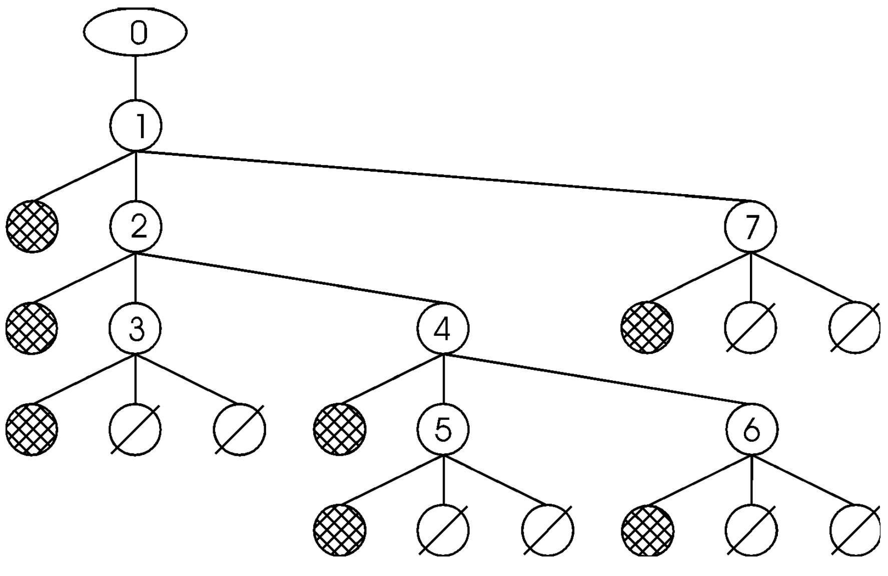

Fig.2. Example of optimization graph in the bisection method

from the set to be analyzed further, and remained unexplored domain is represented in standard form $M = P \bigcap M^0$ with modified ("contracted") HP.

The same algorithm is used for cutting-off "unpromising" brunch, corresponding to the third subset in the current partition. Therefore, while the cuts in the classical branch and bound scheme use an upper bound for the criterion on a subset unpromising for branching, in new method we obtain the domain, where criterion do not surpass the attained record up to optimization error. If unpromising subset is contained in this domain, then corresponding brunch is cut off, else branching continues. A common feature in both methods is that the ε-optimality criterion is used to terminate the search [10]. This ensures the finiteness of the algorithm in difficult problems. In practice, this allows one to terminate the search in a short time.

The initial data for internal cutting algorithm are as follows: the domain $M = P \bigcap M^0$, defined with altering boundaries of HP M^0; the attained formerly record's value

$$Rec = Rec_{init}$$

(at the first step we set $Rec = -\infty$); parameter Δf_0; and the constant L_B, which is set greater by an order of magnitude then diameter of original HP (3), for example:

$$L_B = 20 \max_i (\bar{x}_i - \underline{x}_i).$$

The algorithm results in new value of a record $Rec \geq Rec_{init}$ and the decision about uncut domain existence. This domain, if exists, is given as follows:

$$M' = P \bigcap M^{0'};$$

$$M^{0'} \equiv \{x : \underline{x}_i' \leq x_i \leq \bar{x}_i', i = \overline{1,m}\}. \quad (12)$$

Then the boundaries of HP M^0 are modified so that $M^0 = M^{0'}$.

Algorithm 1

1. Using the simplex method the vertex of local maximum x_A is determined in form (5). If $f(x_A) > Rec$, then we set: $Rec = f(x_A)$.
2. The directing vectors for the edges of polyhedron M adjacent to the point x_A are determined. Accordingly to (6), we have $m - 1$ vectors $B_{*i}, i = \overline{1,m}, i \neq I$, with the components:

   $$B_{ki} = 0, \text{ if } k \neq i \text{ and } k \neq I;\ B_{ii} = -B_{Ii} = \Delta_i .$$

   These vectors are renumbered so that numbers i increase from 1 to $m - 1$.
3. The hyperplane (9), cutting from the cone (7) the simplex S over which the criterion function does not exceed $Rec + \Delta f_0$, is constructed:
   3.1. The support vectors of the cutting hyperplane are constructed: for each vector B_{*i} the maximum normalizing coefficient C such that

   $$f(x_A + CB_{*i}) \leq Rec + \Delta f_0 \text{ and } \|CB_{*i}\|_E \leq L_B \tag{13}$$

   is determined. Then renormalizing is done as follows: $B_{*i} = CB_{*i}$.
   3.2. The pseudoinverse B^+ of matrix (11) is calculated: one removes the lower string of a matrix B, inverts the resulting matrix by Gauss method and completes it with zero column.
   3.3. The components of vector b is calculated in the form (10).
4. The presence of uncut domain $M \setminus S$ is verified:
   4.1. The maximum

   $$L_{\max} = \max_{x \in M} b^{\mathrm{T}}(x - x_A)$$

   is determined with simplex procedure.
   4.2. If $L_{\max} \leq 1$, then $M \setminus S = \varnothing$; i.e. the investigation of the domain M finished because $Rec \geq \max f - \Delta f_0$.
5. The boundaries $\overline{x}_i'$ and $\underline{x}_i'$ are determined for HP (12) that contains the domain $M \setminus S$ and has the edges of a least length.
6. One calculates the "contraction" parameter, measuring percentage decrease of a HP size as follows:

   $$d = 100 \max_i \frac{\overline{x}_i - \overline{x}_i' + \underline{x}_i' - \underline{x}_i}{\overline{x}_i - \underline{x}_i}.$$

7. The assignments $\overline{x}_i = \overline{x}_i', \underline{x}_i = \underline{x}_i',\ i = \overline{1,m}$ are carried out.

Explanations to the algorithm:

To i. 3.1: the restriction $f(x) \le Rec + \Delta f_0$ over the simplex S follows from the first inequality of (13) and Theorem 2. The second inequality in (13) provides normal execution if convex functions monotonically decreases on a ray [4, p. 104].

To i. 3.2: the stability of computations of matrix B^+ is ensured by special form of B. By normalization of support vectors, the matrix B is transformed to the form:

$$B^0 = \frac{1}{\sqrt{2}} \begin{bmatrix} -E & O \\ \multicolumn{2}{c}{1\ 1\ \dots\ 1\ \ (I\text{ - th string})} \\ O & -E \end{bmatrix}, \tag{14}$$

where E are identity matrices of a proper order; O are zero rectangle matrices. One easily can see that angle α between any vector-column of B and span of the others satisfies to inequality:

$$|\sin \alpha| \ge \frac{\sqrt{2}}{2},$$

i.e. there is no multicollinearity of support vectors [11] and pseudoinverse may be calculated.

To i. 4.2: the conclusion that $M \setminus S = \varnothing$ is based on Theorems 1 and 3.

To i. 5: the boundaries $\bar{x}_i'$ and $\underline{x}_i'$ are obtained by linear programming algorithm. HP (12) established in this item is thereafter called the contracted HP.

To i. 6: in further computations the parameter d is used for checking whether a repeated cutting would be "promising" after the domain M has been modified. A repetition of the algorithm is inexpedient if $d \sim 1$.

Since Algorithm 1 is an internal procedure in the branch and bound scheme, the termination of analysis of current domain at i. 4.2 means cutting-off the corresponding branch and passing to a new domain M.

After analysis of the first of three subsets selected at the brunching step has been done, we have to solve the problem of maximizing of $f(x)$ over the domain M with downsized HP M^0. Second and third subsets are constructed by dividing of obtained HP into two halves:

$$\begin{aligned} M_{il}^0 = \{x : \underline{x}_k \le x_k \le \bar{x}_k, \text{ for any } k, k \ne i; \text{ either } \underline{x}_i \le x_i \le \frac{\underline{x}_i + \bar{x}_i}{2}, \text{if } l = 1, \\ \text{or } \frac{\underline{x}_i + \bar{x}_i}{2} \le x_i \le \bar{x}_i, \text{if } l = 2\},\ i = \overline{1, m}, l = 1,2. \end{aligned} \tag{15}$$

Respectively, second and third subsets have the forms $P \cap M_{i_0 l_1}^0$ and $P \cap M_{i_0 l_2}^0$.

This feature explains the name "bisection algorithm" for Algorithm 2 presented below. The variable N in the later algorithm is a number of recursion. The termination of investigation of the domain M, pointed in the item 4.2 of Algorithm 1, means the operations:

If $N > 0$, then go to i. 9 of Algorithm 2 (see below);

If $N > 0$, then finish: the problem (1)—(3) is solved. The maximum of criterion is Rec.

In further description we use a conception of a stack, i.e. the memory space where some values are saved before the recursion. After the recursion lately saved values are extracted from the stack to reset corresponding variables.

Algorithm 2

```
1. Set $N = 0$.
2. Set $Rec = -\infty$.
3. Execute Algorithm 1.
4. If d > 3 then go to i. 3.
5. Set $N = N + 1$.
6. One determines the number $i_0$, specifying the partition of HP $M^0$ into the halves
   $M^0_{i_0 l}, l = 1,2$, and calculates the numbers $l_1$, $l_2$ of the halves to be analyzed in first
   and second turn, respectively.
7. The boundaries of HP $M^0_{i_0 l_2}$ are saved in the variables $\bar{x}^1_i, \underline{x}^1_i, i = \overline{1,m}$ (in particular,
   $\bar{x}^1_i = \bar{x}_i$, $\underline{x}^1_i = \underline{x}_i$ if $i \neq i_0$).
8. Maximization of the function $f(x)$ inside HP $M^0_{i_0 l_1}$:
   8.1. One of the bounds $\bar{x}_{i_0}$ or $\underline{x}_{i_0}$ is modified in accordance with (15) where $l$
        $= l_1$ and $i = i_0$.
   8.2. The contraction procedure is executed, i.e. execute the stand-alone i. 5 of
        Algorithm 1 (the required vector $b$ and vertex $x_A$ were calculated before in i.
        3 of the current algorithm).
   8.3. Set $\bar{x}_i = \bar{x}_i'$, $\underline{x}_i = \underline{x}_i'$, $i = \overline{1,m}$; $Rec1 = Rec$.
   8.4. Put the values $\bar{x}^1_i, \underline{x}^1_i, i = \overline{1,m}; b_i, i = \overline{1,m}; Rec1$ into the stack.
   8.5. Go to i. 2 (recursion).
   Remark. The transition to the next item is possible only from i. 4.2 of Algorithm
1 and means the termination of the later recursion.
9. Data preparations for the maximization inside HP $M^0_{i_0 l_2}$:
   9.1. The variables $\bar{x}^1_i, \underline{x}^1_i, i = \overline{1,m}; b_i, i = \overline{1,m}; Rec1$ recover its value from the
        stack.
   9.2. If $Rec1 > Rec$ then set $Rec = Rec1$.
   9.3. Set $\bar{x}_i = \bar{x}^1_i$, $\underline{x}_i = \underline{x}^1_i$, $i = \overline{1,m}$.
10. Maximization of the function $f(x)$ inside HP $M^0_{i_0 l_2}$:
   10.1. Set $N = N - 1$.
   10.2. Go to i. 3.
```

4. Finiteness of the algorithm and securing of an operating speed

Now we shall establish that, with a certain rules of setting of number *i* in i. 6 of Algorithm 2, the length of a branch beginning in the rooted vertex of the optimization graph would have upper bound, so the number of operations of Algorithm 2 would be finite. In what follows, the Euclidean norm of a matrix (a vector) is denoted by $\|\cdot\|_E$.

<u>Theorem 4.</u> Let a convex function *f*(*x*) is uniformly continuous on the domain (4). Then there exists a value δ > 0 such that, for any plane polyhedron $M = P \cap M^0$ and for any vertex x_A obtained by i. 1 of Algorithm 1, all points $x \in M$ satisfying to inequality

$$\|x - x_A\|_E \le \delta \tag{16}$$

belong to simplex *S*, constructed by i. 3 of mentioned algorithm.

Proof. Uniform continuity means that there exists $\varepsilon' > 0$ such that on the domain *U* holds

$$|f(x') - f(x'')| \le \frac{1}{2}\Delta f_0,$$

as soon as

$$\|x' - x''\|_E \le \varepsilon'.$$

But for any normalizing coefficient *C* determined in i. 3.1 of Algorithm 1, both points x_A and $x_A + CB_{*i}$ belong to *U*, and if the first inequality in (13) becomes an equality, then after normalization we obtain $\|B_{*i}\|_E \ge \varepsilon'$. So in general case we have for normalized vectors B_{*i}:

$$\|B_{*i}\|_E \ge \varepsilon \equiv \min(\varepsilon', L_B).$$

Introduce the matrix, composed of the norms of vectors B_{*i} as follows:

$$N_B = \operatorname{diag}\left\{\pm\|B_{*1}\|_E, \ \dots \ , \pm\|B_{*\,m-1}\|_E\right\}.$$

Here the signs of diagonal entries are chosen so that $B = B^0 N_B$, where B^0 is the matrix from (14). Vector λ with the components defined by (7) for given $x \in M$ evidently has the form $\lambda = N_B^{-1}(B^0)^+(x - x_A)$. Hence, for any point $x \in M$ satisfying to inequality (16), it follows:

$$\sum_{i=1}^{m-1}\lambda_i \le \sqrt{m-1}\|\lambda\|_E \le \sqrt{m-1}\|N_B^{-1}\|_E \|(B^0)^+\|_E \|(x - x_A)\|_E \le$$
$$\le (m-1)\varepsilon^{-1}\|(B^0)^+\|_E \delta \le (m-1)\varepsilon^{-1} Q_m \delta,$$

where $Q_m = \max\limits_{\underline{I=1,m}} \|(B^0)^+\|_E$.

Thus, we can find a value δ, for which the inequality (8) is fulfilled with $C = 1$, i.e. in account of Theorem 3 we establish that vector x belongs to the simplex S.

The theorem has been proved.

It is evident that, if procedure for choosing number i in Algorithm 2 secures decrease of diameter of HP M^0 to zero for infinite brunching (as $N \to +\infty$), then there exists a number N_0 such that, for any $N \geq N_0$ and for $x \in M$, the inequality (16) is fulfilled. In view of Theorem 4, this means that corresponding brunch will be cut off at the step i. 4 of Algorithm 1, and, therefore, the bisection algorithm will have been executed in finite number of steps.

The efficiency of brunch-and-bounds method depends on the way of brunching, in our case on the rule for choosing the indices i_0, l_1, l_2 in i. 6 of the bisection algorithm. Next choosing rule is recommended.

For a given HP M^0 and cutting hyperplane (9), we associate with each polyhedron $M_{il} \equiv P \cap M_{il}^0$ the contracted HPs (12). Denoting their volume by

$$V_{il} = \prod_{i=1}^{m} (\bar{x}_i' - \underline{x}_i'),$$

we set

$$i_0 = \arg\max_i (\bar{x}_i - \underline{x}_i),\ l_2 = \arg\min_{l=1,2} V_{i_0 l}.$$

Given criterion ensures that the halves of HP have a minimum diameter after the partition. Moreover, the half with a smaller volume is inspected in second turn. Thus, on one hand, the maximum length of brunches of optimization graph is bounded, and, on other hand, unpromising brunches should be cut effectively enough.

5. The outcome of trials

The computational results for different problems (1)—(3) let to draw some conclusions about performance of the bisection method. These conclusions are presented below and elucidated by following testing example:

$$f(x) = \sum_{i=1}^{m} \left[(m-i+1)x_i + \frac{\sqrt{i}}{m} x_i^2 \right];$$

$$\underline{x}_i = 0, \bar{x}_i = w = 1,\ i = \overline{1,m}; \quad A = \frac{m}{2}; \Delta f_0 = 0.5 \tag{17}$$

(parameters A and w have been varied for some tests).

All computational results have been obtained on AT-compatible computer, the processor is IDT WinChip C6 (200 MHz).

1. For a given dimension m the volume of computations in the bisection method strongly depends on parameters $\underline{x}_i, \bar{x}_i, A, \Delta f_0$.

In confirmation to this conclusion in Fig. 3 we plot on the diagram a total operating time versus parameter A for $m = 20$ and $m = 40$. All points, except the one indicated, belong to approximation curves. In comparison we plot the dependence of total number of brunching vertices versus A. In Fig. 4 it is plotted the relation of total operating time versus the range $w = \bar{x}_i - \underline{x}_i$ with a given $m = 40$ and $A = 20$. For a values $w > 1.2$ one can reveal the decrease of approximation function, that corresponds to a curves in Fig. 3: the peculiarity of the problem (17) is that optimization time $T(A)$ reaches its maximum for

$$A = 0.5\sum_{i=1}^{m}\left(\bar{x}_i + \underline{x}_i\right)$$

and rapidly decreases when A deviates from this value. It evokes the decrease of an operating time under the essential increase of w and fixed A.

2. In the problems (1)—(3) of a certain class, the optimization time in the bisection method increases with increase of *m* slower then in the total overselection method.

We mean that in new method for a certain class of problems the optimization time is approximated with a relation $T \sim (m-c)^{\alpha}$, where $\alpha > 0$, while in the total overselection method is typical the dependence $T \sim a^m$, $a > 1$. The total overselection method has been chosen for comparison, because only this method is fit when $m \sim 20$.

The results of the test problem solution with both methods being compared are plotted in Fig. 5. The smoothing function takes the next form in the total overselection method:

$$T = 1.807 \cdot 10^{-5} \cdot 2.154^{m} \text{ min};$$

and in the bisection method, respectively:

$$T = 5.37 \cdot 10^{-3} \cdot (m - 10)^{3.04} \text{ min}.$$

For $m < 14$ the total overselection method is advantageous, but the fast growth of vertices number n_M in the feasible HP as a function of m slows down catastrophically optimization process for $m > 21$. It should be noted though that, for a given m, the number n_M strongly depends on the parameters of the feasible domain. Thus, Conclusion 2 is not universal.

When constructing the relations 1 and 2 in Fig. 5, we haven't revealed any difference between the maximization results of bisection method and total overselection method. Besides, the point x_A obtained with a first run of a simplex method (i. 1 in Algorithm 1) turned out to be globally optimal. Similar situation was observed in the applied problem of performance control of missile samples [6, 12], thanks to that it is possible to interpret the locally optimal solution as an approximation to the globally optimal one and to gain the operation speed [12]. However, the cases of non-coincidence of local and global solutions were found both in applied problems and test problems of type (17) (the later for especial value A).

In Fig. 6 it is plotted on the diagram the used memory RAM versus dimension

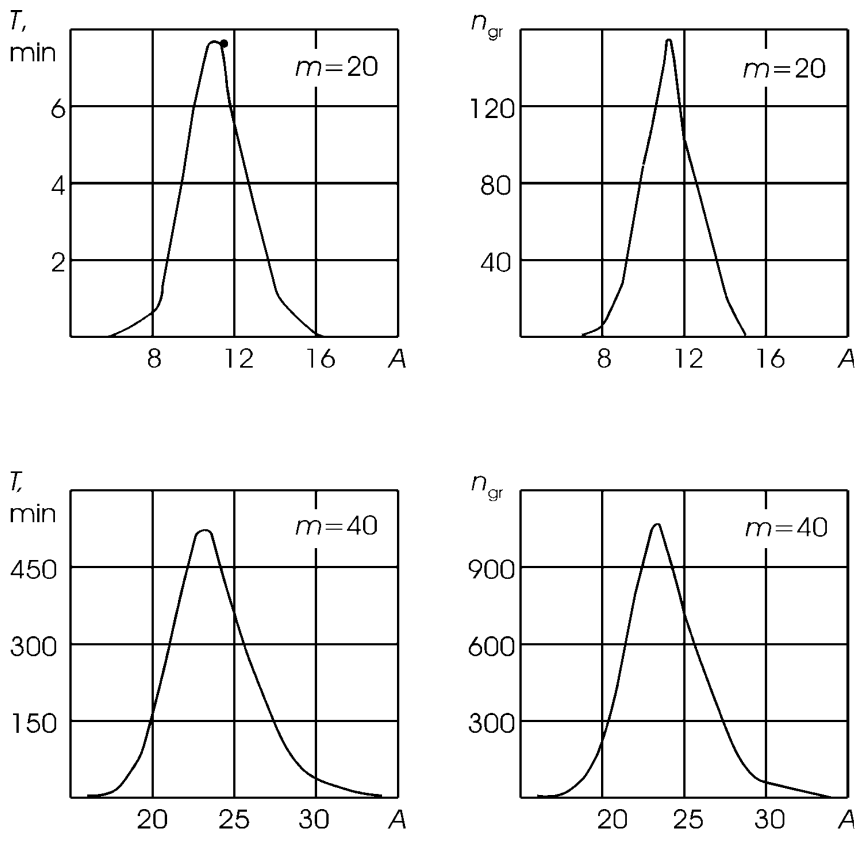

Fig. 3. The plots for a volume of computations in the bisection method versus parameter *A* of the test problem:
T — total operating time, minutes;
n_{gr} — the number of brunching nodes for the optimization graph

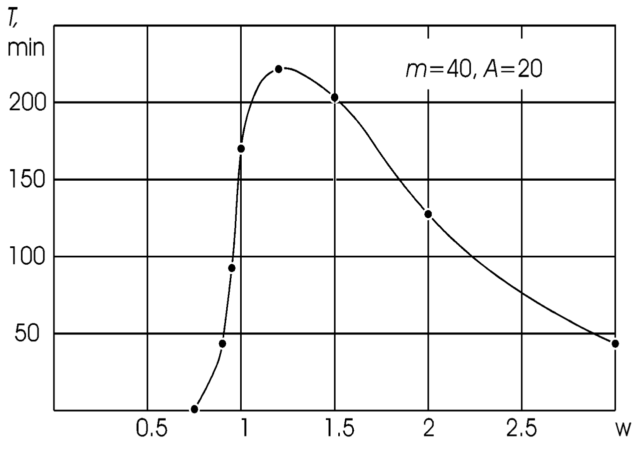

Fig. 4. The plot for an operating time versus range-width *w* in the test problem

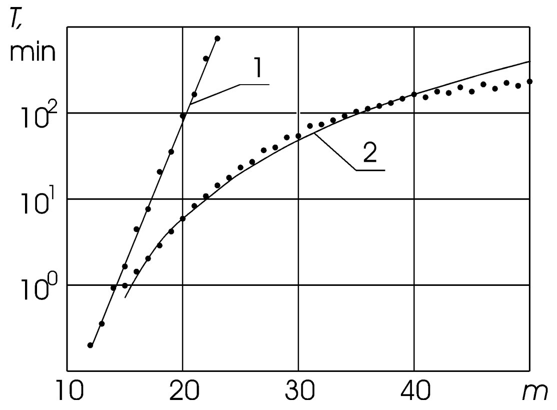

Fig. 5. Optimization time – test problem dimension curves:
1 — total overselection method;
2 — bisection method

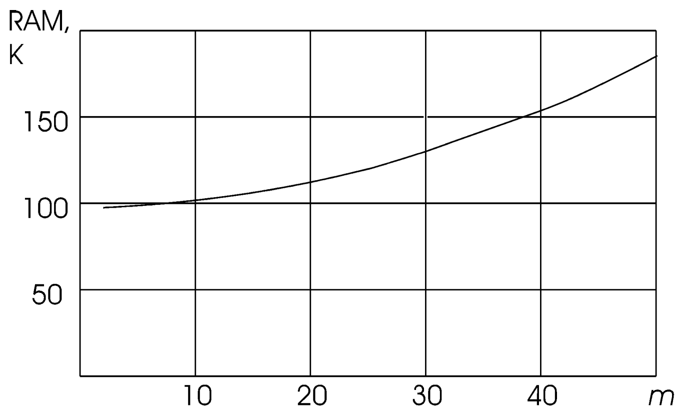

Fig.6. Curve for memory space (KB) versus test problem dimension

of the test problem (17). It may be seen, that it is enough DOS-memory for solving problems with large values *m*.

References

1. Khachaturov V.R., and Utkin S.L. Solving Multiextremal Concave Programming Problems by an Approximation-Combinatorial Method. (In Russian.) Moscow: Computational Center of Academy of Science, USSR, 1988.
2. Utkin S.L., Khachaturov V.R., and Tuy H. Conical Algorithms for Concave Programming Problems and Their Generalizations. USSR, Computational Mathematics and Mathematical Physics, 1988, vol. 28, no. 7, pp. 992–999.
3. Hoang Tuy, Khatchaturov, V., and Utkin, S., A Class of Exhaustive Cone Splitting Procedures in Conical Algorithms for Concave Minimization. Optimization, 1987, no. 18, pp. 791–807.
4. Belousov E.G. Introduction to Convex Analysis and Integer Programming. (In Russian.) Moscow: Moscow State Univ., 1977.
5. Kolmogorov A.N., and Fomin S.V. Elements of the Theory of Functions and Functional Analysis. (In Russian.) Moscow: Nauka, 1981.
6. Rusakov A.I. Checking Hypotheses on the Distribution of System's State Vector by a Series of Regression Experiments in the Multicollinearity Case. RF, Automation and Remote Control, no. 5, 1996, pp. 82–95.
7. Rusakov A.I. The Performance Control of Technical Objects Using Components of Goal Characteristic. RF, Automation and Remote Control, no. 2, 1999, pp. 172–179.
8. Hoang Tuy. Concave Minimization under Linear Constraints with Special Structure. Hanoi: Hanoi Inst. Math., 1983.
9. Hoang Tuy. Concave Programming under Linear Constraints. USSR, Doklady Akad. Nauk SSSR, v. 159, no. 1, 1964, pp. 32–35.
10. Burkov V.N., Lazebnik A.I., and Khranovich I.L. The Branch and Bound Method as a Regular Method for Solving Irregular Mathematical Programming Problems, I,

RF, Automation and Remote Control, 1972, no. 7, pp. 169–177, II, no. 10, pp. 138–147.

11. Demidenko E.Z. Linear and Nonlinear Regression. (In Russian.) Moscow: Finansy i Statistika, 1981.

12. Rusakov A.I. An Improved Reduction Algorithm to Check Hypotheses for the Multicollinear Regression Model. RF, Automation and Remote Control, no. 5, 2001, pp. 94–104.