ISSN 0965-5425, Computational Mathematics and Mathematical Physics, 2009, Vol. 49, No. 2, pp. 246–260. © Pleiades Publishing, Ltd., 2009.
Original Russian Text © Yu.G. Evtushenko, V.U. Malkova, A.A. Stanevichyus, 2009, published in Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 2009, Vol. 49, No. 2, pp. 255–269.

Parallel Global Optimization of Functions of Several Variables

Yu. G. Evtushenko, V. U. Malkova, and A. A. Stanevichyus
Dorodnicyn Computing Center, Russian Academy of Sciences, ul. Vavilova 40, Moscow, 119333 Russia
e-mail: evt@ccas.ru, malkova@ccas.ru, stanev@ccas.ru

Received June 3, 2008;
in final form, September 19, 2008

Abstract—On the basis of the method of nonuniform coverings, a parallel method for the global optimization of Lipschitzian functions is developed. This method is implemented in C–MPI for the global minimization of functions whose gradient satisfies the Lipschitz condition. The performance of the algorithm is demonstrated using the calculation of the structure of a protein molecule as an example.

DOI: 10.1134/S0965542509020055

Keywords: global optimization, method of nonuniform coverings, Lipschitzian functions, parallel computing, multiprocessor systems, Morse function for an atomic cluster energy.

1. INTRODUCTION

In the overwhelming majority of optimization problems, the global optimum must be found. However, due to the great difficulty of this problem, only local solutions are usually determined. Modern multiprocessor computers considerably extend the capability of solving global optimization problems by using parallel algorithms. A survey of the extensive literature on global optimization can be found in [1]. Parallelization of computations is thoroughly studied in [2].

In [3], the method of nonuniform coverings for the global optimization of Lipschitzian functions was proposed and implemented in Algol-60. This method was further elaborated in [4, 5]. If the Lipschitz constant is known, the method of nonuniform coverings yields a guaranteed solution; however, only problems of low dimension can be solved in practice. Numerical results show that the efficiency of this method considerably depends on the Lipschitz constant, which is usually not known a priori; therefore, too high upper bounds are used, which considerably complicates the computations. At the same time, there are domains in many problems in which the Lipschitz constant is not large and the use of the maximal constant is not reasonable. Below, we consider the minimization of functions of three types:

1. functions satisfying the Lipschitz condition,

2. functions whose gradient satisfies the Lipschitz condition,

3. functions whose Hessian satisfies the Lipschitz condition.

For these types of functions, the general scheme of the nonuniform covering method is elaborated with the use of local optimization.

A parallel implementation of the nonuniform covering method and some numerical results for the functions satisfying the Lipschitz condition were described in [6]. In distinction from [6], here we focus on the global minimization of the functions whose gradient or Hessian satisfies the Lipschitz condition. Using C and MPI (Message Passing Interface), a program is developed that finds the global extremum of functions whose gradient satisfies the Lipschitz condition.

2. STATEMENT OF THE PROBLEM AND THE GENERAL IDEA OF THE COVERING METHOD

Consider the problem of the global minimization of the function f on the set $X \subset \mathbb{R}^n$. Define

$$f_* = \operatorname{glob}\min_{x \in X} f(x). \tag{1}$$

Let X_* be the set of solutions of problem (1). Define the set of ε-optimal (approximate) solutions of problem (1):

$$X_*^\varepsilon = \{x \in X : f(x) \leq f_* + \varepsilon\}. \tag{2}$$

It is clear that $X_* \subset X_*^\varepsilon \subset X$. In many cases, it is sufficient to find at least one point $x_r \in X_*^\varepsilon$ and use $f_r = f(x_r)$ as the optimal solution f_*. In other words, we want to determine the global minimum accurate to the given ε and find at least one point x_r where this approximate minimum is attained.

Consider the set of n-dimensional points $N_k = [x_1, \ldots, x_k]$, where every $x_i \in X$. At each x_i, we find the value of f and find the incumbent minimum (the minimal value of the function obtained so far) R_i as

$$R_i = \min_{1 \leq j \leq i} f(x_j) = f(x_\alpha), \quad 1 \leq \alpha \leq i. \tag{3}$$

We also find one of the incumbent minimizers x_α ($1 \leq \alpha \leq i$). It follows from (3) that the sequence R_i is a nonincreasing one.

With each point x_i, we associate its neighborhood P_i such that $x_i \in P_i$. For example, we can use as P_i an n-dimensional parallelepiped containing x_i or a ball centered at x_i.

Consider the following sets X_i and their union:

$$X_i = \{x : x_i \in P_i, f(x) \geq R_i - \varepsilon\}, \quad 1 \leq i \leq k, \quad V_k = \bigcup_{i=1}^{k} X_i. \tag{4}$$

The collection of sets $S_k = [X_1, \ldots, X_k]$ covers the set X if

$$X \subset V_k. \tag{5}$$

Theorem 1. *Suppose that the set of admissible points N_k and the corresponding collection of sets S_k satisfy condition* (5). *Then, the incumbent minimizer x_r found from the conditions $R_k = f(x_r)$ ($x_r \in N_k$) belongs to the set X_*^ε.*

Indeed, if the collection of S_k covers X, then every point x_* in X_* belongs at least to one set X_i. Let $x_* \in X_s$ ($s \leq k$). Then, $R_s \geq R_k$, and we obtain, taking into account (3), that

$$f_* = f(x_*) \geq R_s - \varepsilon \geq R_k - \varepsilon = f(x_r) - \varepsilon.$$

According to (2), this implies the inclusion $x_r \in X_*^\varepsilon$.

This theorem conveys the main idea of the method of nonuniform coverings: instead of finding the global minimum on X, one finds the global minima on subsets whose union contains X. Condition (4) can easily be verified by constructing a minorant of $f(x)$ on the sets P_i or X_i. Minorants of functions satisfying the Lipschitz condition are presented in the next section.

In the implementations of this method, the collections S_k and N_k are constructed one after another, and the incumbent value is improved after each calculation of f; therefore, $R_i \leq f(x_i)$ for all i. Assume that f was last calculated at the point c_m, the incumbent minimizer was c_r ($r \leq m$), and the incumbent minimum was R_m. Define the Lebesgue set

$$\mathcal{L}_m = \{x \in P : R_m - \varepsilon \leq f(x)\}. \tag{6}$$

Assume that x_l is a global minimizer of $f(x)$ at the set $\mathcal{L}_m$. Then, $R_m - f(x_l) = f(c_r) - f(x_l) \leq \varepsilon$. Therefore, the global minimization on $\mathcal{L}_m$ can improve the incumbent minimum R_m not greater than by ε. This implies that $\mathcal{L}_m$ is of no interest, and it can be excluded from the search domain X.

If the condition $X \subset V_m$ is not fulfilled after the computations described above, the minimization procedure is continued on the set $W_m = X \backslash V_m$. Assume that $c_{m+1} \in X_{m+1} \subseteq W_m$ is determined and $f(c_{m+1})$ and R_{m+1} are computed. If $f(x) \geq R_{m+1} - \varepsilon$ on X_{m+1}, then the set X_{m+1} can be excluded from the search, and the minimization procedure is continued on $W_m \backslash X_{m+1}$. Otherwise, X_{m+1} is decomposed into several subsets; in each of them, the values of f are computed, the incumbent minimum is recalculated, some subsets are excluded, others are decomposed, and so on. The procedure terminates when the set X is completely covered. The set of incumbent values is a nonincreasing one. On the contrary, the sequence of the Lebesgue sets $\mathcal{L}_m$ is an expanding one; that is, $\mathcal{L}_m \subseteq \mathcal{L}_{m+i}$ for $i \geq 0$.

The Lebesgue set $\mathcal{L}_m$ is the largest if $R_m = f_*$ in (6). However, f_* is not known a priori. For that reason, in order to make the incumbent value as small as possible, it is reasonable to use local minimization methods for f on X. At the points $c_i \in X$ at which it is found that $f(c_i) < R_m$, the incumbent value is improved; thus one may hope that, solving the local minimization problem

$$\operatorname{loc}\min_{x \in X} f(x),$$

while starting from the point c_i, another point $\bar{c} \in X$ will be found at which $f(\bar{c}) < f(c_i)$. Such a trick often considerably speeds up the computations, improves the incumbent value, and enables one to expand the domain that can be excluded from the search in the process of covering the set X.

3. MINIMIZATION OF LIPSCHITZIAN FUNCTIONS

The most difficult part in the implementation of the nonuniform covering method is the determination of at least a certain part of the set $\mathcal{L}_m$. To this end, one has to impose additional constraints on the function f. Assume that a majorant $G(x, y)$ and a minorant $g(x, y)$ for $f(x)$ exist such that

$$G(x, y) \geqslant f(x) \geqslant g(x, y), \quad G(y, y) = g(y, y) = f(y) \tag{7}$$

for all $x, y \in X$.

Define the sets

$$K_i = \{x \in P_i : g(x, c_i) \geqslant R_i - \varepsilon\}, \quad U_k = \bigcup_{i=1}^{k} K_i. \tag{8}$$

It follows from (6) and (7) that $K_i \subseteq \mathcal{L}_m$; therefore, K_i can be excluded from the search domain X. Comparing definitions (4) and (8), we see that $K_i \subset X_i$ and $U_k \subset V_k$. Theorem 1 can be reformulated as follows. If the collection of feasible points N_k and the corresponding sets $K_1, \ldots, K_k$ are such that $X \subset U_k$, then the incumbent minimizer x_r found from the conditions $R_k = f(x_r)$ $(x_r \in N_k)$ belongs to X_*^{ε}.

To construct a covering of X, one must know how to construct minorants. They can be easily found for Lipschitzian functions. We consider three classes of such functions.

1. Assume that the function f satisfies the Lipschitz condition with a constant l; that is, we assume that, for any x and y in X, it holds that

$$|f(x) - f(y)| \leqslant l\|x - y\|. \tag{9}$$

Hence, we obtain the majorant $G(x, y)$ and the minorant $g(x, y)$:

$$G(x, y) = f(y) + l\|x - y\| \geqslant f(x) \geqslant f(y) - l\|x - y\| = g(x, y). \tag{10}$$

Using this minorant, definition (8) can be written in the form

$$K_i = P_i \bigcap H_i, \quad H_i = \{x \in \mathbb{R}^n : \|x - c_i\| \leqslant \rho_{ir}\}, \tag{11}$$

where

$$\rho_{ir} = [\varepsilon + f(c_i) - R_r]/l, \quad 1 \leqslant r \leqslant i \leqslant k. \tag{12}$$

If $\|\cdot\|$ is the Euclidean norm, then H_i is the ball of the radius ρ_{ir} centered at c_i. If $\|\cdot\|$ is the Chebyshev norm, then H_i is the cube centered at c_i with the principal diagonal $2\rho_{ir}$.

The vectors c_i belong to X for all $1 \leqslant i \leqslant k$. According to (3), $f(c_i) \geqslant f(c_r) = R_r$ for $1 \leqslant r \leqslant i$. Therefore, the ball of radius ρ_{ir} given by formula (12) is always greater than or equal to ε/l. Define

$$\xi = 2\varepsilon/l. \tag{13}$$

The quantity ξ is called the minimal diameter of the covering ball. Such a diameter is realized when $f(c_i) = R_r$. For $x \in H_i$, we have

$$f(x) \geqslant \min_{x \in X} g(x, c_i) = f(c_i) - l\max_{x \in H_i}\|x - c_i\| = f(c_i) - l\rho_{ir} \geqslant R_r - \varepsilon.$$

Therefore, the ball H_i can be excluded from the search domain. If the union of the balls H_i $(1 \leqslant i \leqslant k)$ satisfying (11) covers X, then $c_r \in X_*^{\varepsilon}$. The radius of H_i is large if $f(c_i) \geqslant f(c_r)$, and it is small in the domains where $f(c_i) \approx f(c_r)$. Due to this fact, X can be covered by balls of different radii. This idea was first published and implemented in [3].

2. Let the function $f(x)$ be differentiable and its gradient satisfy the Lipschitz condition, that is, there is a constant L such that

$$\|f_x(x) - f_x(y)\| \le L\|x - y\| \tag{14}$$

for any x and y in X. It is easy to show that, if $f(x)$ is twice differentiable, then (14) holds if and only if $\|f_{xx}(x)\| \le L$ for any $x \in X$. It follows from (14) that the concave support minorant $g(x, y)$ and the convex support majorant $G(x, y)$ are determined by

$$g_1(x, y) = f(y) + \langle f_x(y), x - y\rangle - \frac{L}{2}\|x - y\|^2, \tag{15}$$

$$G_1(x, y) = f(y) + \langle f_x(y), x - y\rangle + \frac{L}{2}\|x - y\|^2. \tag{16}$$

This implies the inequality

$$L\|x - y\|^2 \ge 2|f(x) - f(y) - \langle f_x(y), x - y\rangle|. \tag{17}$$

This inequality can be used to check the correctness of the Lipschitz constant L. If (17) is not fulfilled, then either L must be increased or assumption (14) is not true for the function under examination, or conditions (15) and (16) can be used only locally for certain subsets $H_i \subset X$.

We use formulas (15) and (16) to find the global minimum of f on X. We will seek the minimum of the minorant and majorant with respect to x. For $y \in X$, we find $\hat{x}_1$ and $\hat{x}_2$ such that

$$\hat{x}_1 \in \operatorname*{argmin}_{x \in X}\left[\langle f_x(y), x - y\rangle - \frac{L}{2}\|x - y\|^2\right], \tag{18}$$

$$\hat{x}_2 \in \operatorname*{argmin}_{x \in X} G_1(x, y). \tag{19}$$

The second problem is much easier because $G_1(x, y)$ is convex. It is clear that

$$f(\hat{x}_1) \le \min_{x \in X} f(x) \le f(\hat{x}_2). \tag{20}$$

If $\hat{x}_2$ is an interior point of X, then

$$\hat{x}_2 = y - \frac{f_x(y)}{L}, \quad f(\hat{x}_2) = f(y) - \frac{\|f_x(y)\|^2}{2L}.$$

In the numerical implementation of the method, the minimum of $f(x)$ was sought on the segments connecting the point y with $\hat{x}_1$ and the point y with $\hat{x}_2$. In many cases, this procedure improved the incumbent minimum.

It follows from (10) and (15) that, for all x and y in X, it holds that

$$f(x) \ge f(y) + \langle f_x(y), x - y\rangle - \frac{L}{2}\|x - y\|^2. \tag{21}$$

The condition $f(x) \ge R_r - \varepsilon$ is fulfilled if x satisfies the condition

$$\frac{L\|x - y\|^2}{2} + \langle f_x(y), x - y\rangle \le \varepsilon + f(y) - R_r. \tag{22}$$

This set can be excluded from X in the numerical global minimization.

When X is a parallelepiped, the minimization problem for the minorant has an analytical solution that is presented in the next section.

3. Let f be a twice differentiable function whose Hessian satisfies the Lipschitz condition with the constant M:

$$\|f_{xx}(x) - f_{xx}(y)\| \le M\|x - y\|. \tag{23}$$

According to (14), the support minorant and majorant of $f(x)$ in this case have the form

$$g_2(x, y) = f(y) + \langle f_x(y), x-y\rangle + \frac{1}{2}(x-y)^{\mathrm{T}} f_{xx}(y)(x-y) - \frac{M}{6}\|x-y\|^3,$$

$$G_2(x, y) = f(y) + \langle f_x(y), x-y\rangle + \frac{1}{2}(x-y)^{\mathrm{T}} f_{xx}(y)(x-y) + \frac{M}{6}\|x-y\|^3.$$

In distinction from (15) and (16), these formulas provide approximations of the second order of accuracy for $f(x)$. In the general case, both functions are nonconvex here. The minimization of the majorant $G_2(x, y)$ is studied in [7].

Let $\gamma(y)$ be the minimal eigenvalue of the matrix $f_{xx}(y)$. Define $\Delta = \|x - y\|$. Then,

$$g_2(x, y) \geqslant f(y) + \left[-\|f_x(y)\| + \frac{1}{2}\gamma(y)\Delta - \frac{M}{6}\Delta^2\right]\Delta.$$

The condition $f(x) \geqslant R_r - \varepsilon$ is fulfilled inside the ball centered at y with the radius Δ satisfying the condition

$$\Delta^3 + \frac{3\Delta}{M}[\gamma(y)\Delta - 2\|f_x(y)\|] \leqslant \frac{6}{M}[\varepsilon + f(y) - R_r], \tag{24}$$

where $f(y) \geqslant R_r$.

We represent the symmetric matrix $f_{xx}(y)$ as the difference of two matrices:

$$f_{xx}(y) = f^1_{xx}(y) - f^2_{xx}(y).$$

Here, $f^1_{xx}(y)$ and $f^2_{xx}(y)$ are positive definite matrices. The minorant $g_2(x, y)$ is reduced to the d.c. form, that is, to the difference of two convex functions:

$$g_2(x, y) = g_{21}(x, y) - g_{22}(x, y).$$

Here,

$$g_{21}(x, y) = f(y) + \langle f_x(y), x-y\rangle + \frac{1}{2}(x-y) f^1_{xx}(y)(x-y),$$

$$g_{22}(x, y) = \frac{1}{2}\left((x-y)^{\mathrm{T}} f^2_{xx}(y)(x-y) + \frac{M}{6}\|x-y\|^3\right).$$

An extensive list of references concerning d.c. functions (difference of two convex functions) and methods of their optimization can be found in [8].

The approach discussed here can be extended to the case when the jth derivative of f satisfies the Lipschitz condition. As j increases, the approximation of the majorant and minorant of f becomes more accurate, which improves the efficiency of the nonuniform covering method; however, the complexity of the calculations increases because high-order derivatives of $f(x)$ must be found.

The techniques presented above guarantee that the global minimum will be found provided that the Lipschitz constants are known. The main disadvantage of this approach is that certain upper bounds have to be used as Lipschitz constants, which considerably complicates the calculations, especially in the domains where the gradient of $f(x)$ is small. The simplest solution is to assume that a particular Lipschitz constant exists in each ball H_i. To this end, (9) and (14) are replaced with the conditions

$$l_i = \max_{y \in H_i}\max_{x \in H_i}|f(x) - f(y)|/\Delta,$$

$$L_i = \max_{y \in H_i}\max_{x \in H_i}|f_x(x) - f_x(y)|/\Delta.$$

Finding exact constants using these formulas can be more difficult than the global minimization of $f(x)$ on H_i. For this reason, one may use approximate estimates. In particular, if the diameter of H_i is sufficiently small, one may assume that $l_i \simeq \|f_x(c_i)\|$ and $L_i \simeq \|f_{xx}(c_i)\|$, where c_i is the center of H_i.

4. THE SEQUENTIAL COVERING ALGORITHM

When the method is implemented in software, an additional simplifying assumption is made. Namely, we assume that the feasible set X is an n-dimensional parallelepiped with the faces parallel to the coordinate planes:

$$X = \{x \in \mathbb{R}^n, a \leqslant x \leqslant b\}.$$

Here and in what follows, the inequality $a \leqslant x$ means that $a^j \leqslant x^j$ for all $1 \leqslant j \leqslant n$.

In the calculations, we use the additional vectors $a_i, b_i \in \mathbb{R}^n$ and the rectangular parallelepipeds with the faces parallel to the coordinate planes generated by them: $P_i = \{x \in \mathbb{R}^n : a_i \leqslant x \leqslant b_i\}$.

We assume that $a \leqslant a_i < b_i \leqslant b$. Thus, all the parallelepipeds $P_i \subseteq X$. The center $c_i = (a_i + b_i)/2$ of P_i is used as the vector y. Then, the radius of the ith parallelepiped is

$$\Delta_i = \|c_i - a_i\| = \|c_i - b_i\| = \frac{1}{2}\|a_i - b_i\|.$$

The Chebyshev norm of the vector $d_i = b_i - a_i$ determines the length of the longest edge of P_i:

$$w_i = \|d_i\|_\infty = \max_{1 \leqslant j \leqslant n} d_i^j = 2\Delta_i.$$

Denote by q_i the minimum of the minorant of $f(x)$ on P_i. If $f(x)$ satisfies Lipschitz condition (9) everywhere in P_i with the constant l_i, then, according to (10), we have

$$q_i = \min_{x \in P_i} g(c_i, x) = f(c_i) - l_i\Delta_i. \tag{25}$$

If the gradient of $f(x)$ satisfies condition (14) everywhere in P_i with the constant L_i, then q_i is found by minimizing minorant (15) on P_i; that is,

$$q_i = \min_{x \in P_i}\left[f(c_i) + \langle f_x(c_i), x - c_i\rangle - \frac{L_i}{2}\|x - c_i\|^2\right].$$

For all $x \in P_i$, we have

$$a_i \leqslant x_i \leqslant b_i, \quad \frac{(a_i - b_i)}{2} \leqslant x_i - c_i \leqslant \frac{(b_i - a_i)}{2}.$$

The minimum of the minorant with respect to x_i^j on P_i is attained on the boundary at $x_*^j = b_i^j$ if $\dfrac{\partial f(c_i)}{\partial x^j} < 0$ and at $x_*^j = a_i^j$ otherwise. In both cases, $|x_*^j - c_i^j| = b_i^j - c_i^j$ for $1 \leqslant j \leqslant n$ and

$$q_i = f(c_i) - \langle |f_x(c_i)|, b_i - c_i\rangle - \frac{L_i}{2}\|b_i - c_i\|^2.$$

Here, $|f_x(c_i)|$ denotes the n-dimensional vector in which the jth component is $\left|\dfrac{\partial f(c_i)}{\partial x^j}\right|$. This yields the following lower bound on the minimum of the minorant on the parallelepiped P_i:

$$q_i \geqslant f(c_i) - Q_i\|b_i - c_i\| - \frac{L_i}{2}\|b_i - c_i\|^2, \quad \text{where} \quad Q_i = \|f_x(c_i)\|. \tag{26}$$

If $f(x_*) < R_r$, we set $R_r = f(x_*)$. According to (26), the inequalities

$$f(x) \geqslant q_i \geqslant R_r - \varepsilon \tag{27}$$

hold true if

$$\|b_i - c_i\|^2 + \frac{2}{L_i}Q_i\|b_i - c_i\| \leqslant \varphi_{ir}, \quad \text{where} \quad \varphi_{ir} = \frac{2}{L_i}[\varepsilon + f(c_i) - R_r] \geqslant \frac{2\varepsilon}{L_i}.$$

Consider the ball S_i

$$S_i = \{x : \|x - c_i\| \leqslant \zeta_{ir}\},$$

with the radius ζ_{ir}, where

$$\zeta_{ir} = \sqrt{\theta_i^2 + \varphi_{ir}} - \theta_i, \quad \theta_i = \frac{Q_i}{L_i}.$$

For $Q_i = 0$, we have $\zeta_{ir} = \sqrt{\varphi_{ir}}$; in this case, the diameter of the ball is minimal:

$$\xi = 2\sqrt{\frac{2\varepsilon}{L_i}}. \tag{28}$$

For $L_i = 0$, we have $\zeta_{ir} = \varepsilon + f(c_i) - R_r \geqslant \varepsilon$.

If conditions (9) or (14) are fulfilled, the minimal diameter of the covering ball is determined by (13) and (28), respectively. If condition (23) is fulfilled, ξ is given by (24).

If

$$\Delta_i \leqslant \zeta_{ir}, \tag{29}$$

then $P_i \subset S_i$, the parallelepiped P_i can be excluded from P, and the search for the minimum can be continued on the set $P \backslash S_i$. If inequality (29) is not fulfilled, the parallelepiped is sequentially subdivided by cutting it perpendicularly to some edge.

Cutting P_i perpendicular to the longest edge seems to be the most simple and natural. However, if the range of the values of the gradient components is large, an analog of scaling can be used. If $f(x)$ is differentiable, the edge with the index α for which

$$\alpha = \arg \max_{1 \leqslant j \leqslant n} (b_i^j - c_i^j) \left| \frac{\partial f(c_i)}{\partial x^j} \right| \tag{30}$$

can be cut. Such a variant of subdividing the parallelepiped was proposed in [9]. In the particular case when all the components of the gradient are identical, α is the index of the longest edge.

If the differentiability is not used, condition (30) can be replaced by the following one:

$$\alpha = \arg \max_{1 \leqslant j \leqslant n} \left| f(c_{i2}^j) - f(c_{i1}^j) \right|. \tag{31}$$

Here, the vectors c_{i2}^j and c_{i1}^j have the same components as the vector c_i except for the jth components, which are c_{i2}^j and c_{i1}^j for a_i^j and b_i^j, respectively.

The software implementation of the method of nonuniform coverings is performed almost identically for different Lipschitzian functions. Only the bounds on the minimum q_i and the minimal diameters of the covering balls are found differently. The algorithm of covering the set X by the parallelepipeds P_i is the same.

If $q_i \geqslant R_i - \varepsilon$, then P_i can be excluded because the global minimum on it cannot improve the incumbent minimum R_m more than by ε; the search is continued on the set $X \backslash P_i$. Otherwise, if $q_i < R_m - \varepsilon$, P_i is cut in two, for example, perpendicularly to the longest edge, and the search is continued in these two parts. If the principal diagonals of one or both parts are less than ξ, then $f(x) \geqslant q_i = f(c_i) - \varepsilon$ on them and these parts are excluded from the search domain. At the centers of the remaining parts, the values of $f(x)$ are computed. If possible, the incumbent minimum is improved, and the coverage condition of both parts is checked. The covered parts are excluded from the search domain. If both parts are excluded, P_i is also excluded. Otherwise, the subdivision is continued until P_i is covered. Various techniques for covering the parallelepiped X can be used. For example, the layer-by-layer covering using Algol-60 recursive procedures was implemented in [3]. This variant was used to solve global optimization problems of low dimension. Below, we use the branch-and-bound method (as in [4]) for covering X.

With each parallelepiped P_i, we associate the set $S_i = (c_i, d_i, q_i)$. The collection of S_i for the set of uncovered parallelepipeds B_m will be called the list of sets, and we denote it by $S = \{S_1, \ldots, S_m\}$. The notation $S = \varnothing$ means that the list S is empty.

From the list S, we select a parallelepiped to start the covering process; it is called the working parallelepiped. We used the following three selection rules.

MINQ rule. The parallelepiped P_s is selected for which $q_s = \min\limits_{1 \leqslant i \leqslant m} q_i$.

FIFO rule. The first parallelepiped is selected; i.e., $s = 1$.

LIFO rule. The last parallelepiped is selected; i.e., $s = m$.

In the description of the algorithm below, the symbol k denotes the number of the iteration of the main loop.

Initial operations. Set $k = 1$, $m = 1$, and $P_1 = P$. Set $\varepsilon > 0$, the Lipschitz constant (l, L, or M), and a point $x_0 \in P$. Calculate c_1, w_1, $f(c_1)$, $f(x_0)$, q_1, and determine $\tilde{R} = \min\{f(c_1), f(x_0)\}$. Set $N_1^{(1)} = \{c_1\}$, $B_1^{(1)} = \{P_1\}$, $S_1 = \{c_1, w_1, q_1)$, and $S^{(1)} = \{S_1$. If $\tilde{R} = f(c_1)$, then use c_1 as the incumbent minimizer; otherwise, use x_0. If $q_1 \geqslant \tilde{R} - \varepsilon$, then terminate the calculation.

Main loop. Repeat the following steps 1–5 while $S(k) \neq \varnothing$, that is, until the set X is completely covered.

Step 1. Using one of the selection rules, choose the working parallelepiped P_s.

Step 2. Find the longest edge index t in P_s: $w_s = d_s^t$.

Step 3. Divide P_s into two parts with respect to the component t thus creating two new parallelepipeds P' and P''. Their centers, the lengths of the longest edges, and the minimal values of the minorants are denoted by c', w', q' and c'', w'', g'', respectively. Then, eliminate P_s from the set $B_m^{(k)}$, remove S_s from the list $S^{(k)}$, and set $m = m - 1$.

Step 4. Calculate

$$\tilde{R} = \min\{f(c'), f(c'')\}. \tag{32}$$

If $\tilde{R} \geqslant R^{(k)}$, then set $R^{(k+1)} = R^{(k)}$. If $q' < R^{(k+1)} - \varepsilon$ and $w' \geqslant \xi$, then add $S' = (c', w', q')$ to the list of collections and set $m = m + 1$. If $q'' < R^{(k+1)} - \varepsilon$ and $w'' \geqslant \xi$, then add $S'' = (c'', w'', q'')$ to the list of collections and set $m = m + 1$. If $\tilde{R} < R^{(k)}$, then set $R^{(k+1)} = \tilde{R}$. The incumbent minimizer x_r is set either to the point c' or to c'' depending on where the minimum in (32) is attained. Every set S_i in $\{S_i\}_{1 \leqslant i \leqslant m}$ satisfying the condition $q_i \geqslant R^{(k+1)} - \varepsilon$ is removed from this list. Renumber the new list $\{S_{i1}, \ldots, S_{im}\}$ and set $S^{(k+1)} = \{S_j\}_{1 \leqslant j \leqslant m}$.

Step 5. Set $k =: k + 1$.

Note that the principal diagonal of the parallelepiped obtained using the subdivision procedure described above cannot shorter than d. Taking this fact into account, its is easy to prove the following result.

Proposition 1. *Let the function f in problem* (1) *or its first- or second-order derivatives satisfy the Lipschitz condition on the n-dimensional rectangular parallelepiped P. Then, the algorithm described above yields the incumbent minimizer* $x_r \in X_*^\varepsilon$ *using a finite number of computations of f.*

After the computations are completed, the set P is completely covered by the parallelepipeds P_i, and the incumbent minimizer x_r is an ε-solution of problem (1).

This algorithm can be considerably accelerated by using local minimization procedures. To obtain the results presented below, the conjugate gradient method with projecting on the parallelepiped P was used for the local minimization. The software implementation of the algorithm was performed by N.I. Grachev.

The presented algorithm can be interpreted as the branch-and-bound method (see [10]): at step 1 the set for branching is selected; at step 2 the branching procedure is chosen; at step 3 the branching is performed; and, at step 4 the bounds q' and q'' corresponding to the new sets that are candidates for branching are calculated, the incumbent minimum R is updated, and certain sets are excluded from the search domain (if the incumbent minimum was improved).

At each step of the method, a parallelepiped belonging to the current collection is divided into two parallelepipeds by a plane that is parallel to a coordinate plane. The process of bisection can be interpreted as the growth of a binary tree. The vertices of this tree are associated with the parallelepipeds obtained in the initial partitioning. The edges connect the given parallelepiped with the parallelepipeds obtained by its subdivision. The parallelepipeds corresponding to the leaf vertices of the tree form the current set of parallelepipeds. Some leaf vertices can be removed from this set using the pruning rule (see step 4).

For brevity, we will call the current collection of parallelepipeds a *pool* and Steps 1–5 of the main loop will be called an *iteration*.

The efficiency of a particular covering of the feasible set becomes considerably higher if the initial point is chosen properly. Such a choice yields an incumbent minimum that is close to the global minimum to be found. To find the initial value, calculations with deliberately low Lipschitz constants are first performed. In this case, the majorants and minorants can considerably differ from $f(x)$. Inequalities (20) can be violated in this case.

5. DESCRIPTION OF A PARALLEL ALGORITHM

The main idea behind the proposed parallel algorithm is to perform iterations concurrently on several processors that periodically exchange the incumbent minima; the search domains are periodically redistributed between the processors in the course of the computations.

To concurrently perform the iterations, one needs a method for distributing the parallelepipeds between the processors. Assume that there are p processors and there is an initial sequence $B_m = \{P_1, \dots, P_m\}$ of the parallelepipeds P_i belonging to P; for example, this sequence could be preliminary constructed by the same algorithm. This sequence can consist of a single parallelepiped ($m = 1$) or of a collection of s subsequences of parallelepipeds (trees). Let the incumbent minimum R_m and the incumbent minimizer x_r be found using (3).

The distribution of the work between the processors depends on the unit of work of the algorithm. As such a unit, we use the computational work needed to perform an iteration of the sequential algorithm Q times ($Q \geqslant 1$).

Assume that, in addition to the p processors mentioned above, which will be called slaves, there is an additional processor called the *master.* Let us distribute the parallelepipeds in the initial collection $B_m$among the slave processors ($p \geqslant s$).

When performing the iterations, each slave processor maintains its own pool of parallelepipeds to be examined. We say that a processor finished its portion of the work if it performed Q iterations of the sequential algorithm (performed Q bisections) or if its individual pool is empty. The processor that finished its work sends to the master the following data: the number of parallelepipeds in its pool, the minimal bound y for each of them, and the incumbent minimum of f obtained in the course of the calculations. This processor is placed in the waiting queue maintained by the master. A waiting processor is said to be *free* if its individual pool is empty.

Consider the communication between the processors. The exchange of information would be ideal if each processor could send to all the other processors the incumbent minimum found by it, the number of parallelepipeds in its pool, and the minimal bound for each of them. However, too frequent data exchange between the processors can significantly increase the communication overheads and reduce the efficiency of the parallel algorithm as a whole.

In the proposed variant of the algorithm, each slave processor sends to the master the data indicated above in the asynchronous mode, i.e., independently of the other processors. Using these data, the master processor improves the incumbent minimum R and communicates it to the waiting processors, which will use this minimum to remove parallelepipeds from their collections. The master processor finds among the waiting processors the ones that contain only a single parallelepiped in their individual collection and instructs them to perform the main loop Q times using R as the incumbent minimum. The same instructions are given to the other waiting processors if there are no free processors at the moment.

If there are still undistributed parallelepipeds from the initial collection, their subsequences are sent to the detected free processors.

If all the parallelepipeds from the initial collection are already distributed, the master processor detects a free slave processor and chooses a processor among the waiting ones; then the master instructs the waiting processor to pass the parallelepiped with the minimum lower bound to the free processor and instructs the free processor to get this parallelepiped.

The master completes the execution of the algorithm when all the slave processors are free.

In the proposed algorithm, the frequency of the communications between the slave processors and the master depends on the parameter Q. As soon as a free processor is detected, the work is redistributed between the processors by passing parallelepipeds. In such a scheme, the processors execute the algorithm asynchronously exchanging information only with the master. As a result, the execution of the algorithm becomes nondeterministic; that is, the sequence of parallelepipeds and the solution may be different for different runs of the same program for the same problem (even on the same set of processors).

Let us describe the parallel algorithm for the interaction scheme between the processors presented above. To exchange data between the processors, we use the explicit message passing. A message sent from a slave processor to the master contains the following information: the number of parallelepipeds in the current pool, the minimal bound q for the parallelepipeds in the current pool, and the individual incumbent minimum $R = f(c)$.

The master processor sends to the slaves the following instruction messages:

—*work*(Q, R) to execute the main loop Q times using R as the incumbent minimum,

—*read*(T) to receive the initial subsequence of parallelepipeds T,

—*take*(i) to receive a parallelepiped from processor i,

—*give*(j) to pass a parallelepiped to processor j,

—*finish* to finish the operation.

Algorithm for the master processor

Step 1. Make all the slave processors free. If the initial sequence contains a single parallelepiped (m = 1), then make the processor p_1 a waiting one and add to its individual pool the single initial parallelepiped; set the incumbent minimum to $R = f(x_0)$, where $x_0 \in P$ is the given initial point; set the bound y to the bound assigned to the initial parallelepiped; and set the parameter $Q \geqslant 1$.

Step 2. If the given operation time specified for the processors is not expired, repeat the following steps 3–7 until all the processors become free.

Step 3. If there are no free processors, send the message *work* to all the waiting processors and clear the list of the waiting processors.

Step 4. Send the message *work* to each processor that has a single parallelepiped in its pool and remove this processor from the list of the waiting processors.

Step 5. While there are free processors and unassigned parallelepipeds in the initial sequence, execute the following loop: send the message *read* to every free processor and remove it from the list of free processors.

Step 6. (Loop for selecting pairs of processors to pass parallelepipeds.) While the lists of the waiting and free processors are not empty, find among the waiting processors the one with the minimal bound, take a processor j from the list of the free ones, remove the selected processors from the corresponding lists, send the message *give*(j) to the former and the message *take(i)* to the latter.

Step 7. Receive the messages sent by the slave processors, add the sending processor to the list of the waiting or free processors depending on the number of parallelepipeds in its pool. Then, update the incumbent minimum taking into account the data obtained from the processors.

Step 8. Send the message *finish* to each slave processor.

Algorithm for the slave processors

Step 1. Until the finish instruction is received, execute the following steps 2–4.

Step 2. Receive the next instruction from the master processor.

Step 3. If the instruction *give* or *take* is received, execute it and then send a message to the master containing the number of parallelepipeds in the pool and the minimal bound y.

Step 4. If the instruction *work* is received, execute the iterations while the individual pool is not empty and less than Q iterations are executed; next, send a message to the master containing the results.

Step 5. If the instruction *read* is received, execute it and send a message to the master containing the number parallelepipeds in the pool and the minimal bound y.

Step 6. If the instruction *finish* is received, complete the work.

6. NUMERICAL RESULTS

For the numerical calculations, we used the Morse function for the energy of atomic clusters consisting of n atoms:

$$f(x_1, \ldots, x_n, \rho) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} [(e^{\rho(1-\|x_i - x_j\|)} - 1)^2 - 1].$$

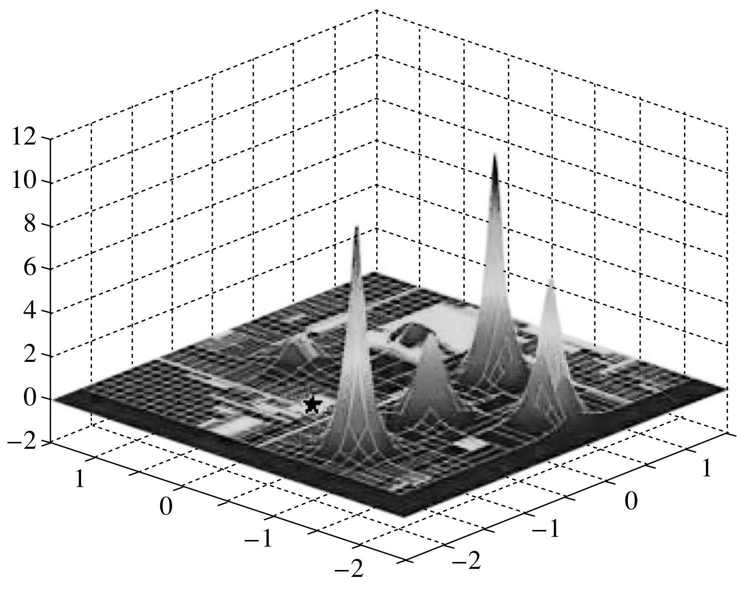

Fig. 1.

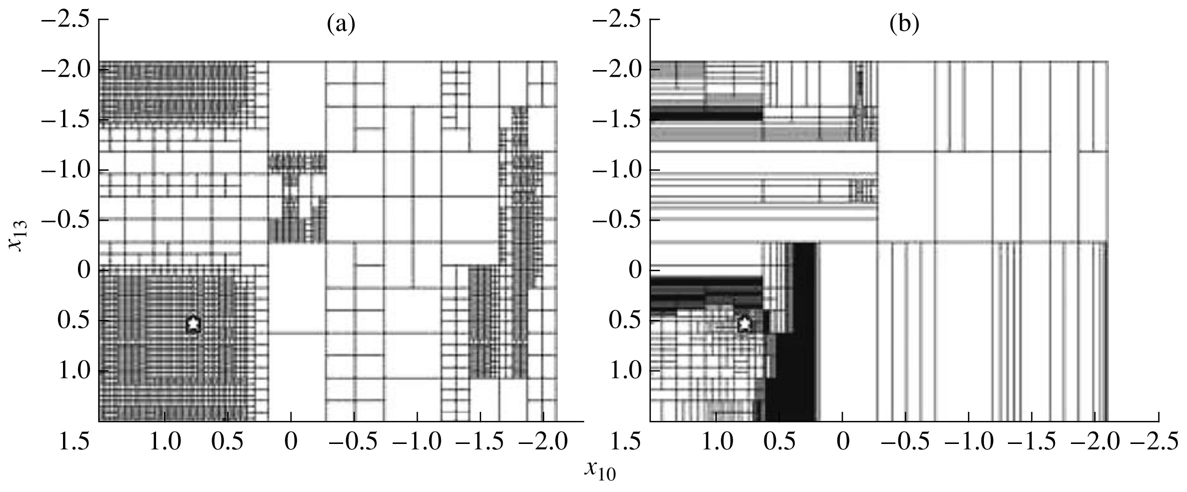

Fig. 2.

Here, ρ is a scalar parameter and $x_i \in \mathbb{R}^3$ and $x_j \in \mathbb{R}^3$ are the vectors of coordinates of the atom's i and j centers, respectively.

We want to find the Cartesian coordinates of n atoms that globally minimize f. The parameter ρ was set to 3, 6, 8, 10, and 14; and the number of atoms was as much as 85 (in this case, the number of scalar variables to be found is 255). Figure 6 shows the structure of a cluster with 85 atoms and $\rho = 6$; the potential energy in this case is –405.246158.

The computations were performed under two assumptions: the function f satisfies the Lipschitz condition with a constant l (strategy I) and its gradient satisfies the Lipschitz condition with a constant L (strategy II). Since l and L are not known a priori, a series of computations was performed: first, with small values of l and L; after determining an incumbent minimum, l and L were gradually increased so as to keep the computation time reasonable.

The computations were performed on the multiprocessing system MVS 100k installed at the Joint Supercomputer Center (JSCC) of the Russian Academy of Sciences [11]. The computations were also performed in the distributed GRID environment at the Dorodnicyn Computing Center of the Russian Academy of Sci-

Table 1

Strategy	The method for choosing an edge	Number of parallelepipeds	Percentage of unreliable estimates
		$f(x_{10}, x_{13})$	
I	*a*	16947	14.22%
II	*a*	7895	0.00%
II	*b*	2097	0.05%
		$f(x_4, x_{12})$	
I	*a*	17327	1.63%
II	*a*	7419	0.05%
II	*b*	3781	0.08%
		$f(x_0, x_3)$	
I	*a*	33761	10.06%
II	*a*	5965	1.15%
II	*b*	4823	2.26%

Table 2

Characteristics	Rule for choosing the working parallelepiped		
	MINQ	FIFO	LIFO
Solution time (s.)	145	49	67
Number of parallelepipeds	200285	78697	113967
Index of the parallelepiped at which the minimum was found	135850	30924	89606
The maximum size of the list (number of parallelepipeds)	12821	12865	22

ences and the Institute for System Programming of the Russian Academy of Sciences using the Globus 4.0.4 technology. The results for the atomic clusters with less than 80 atoms were in very good agreement with the data presented in the Cambridge Cluster Database [12].

To compare the efficiency of the two strategies, we considered the Morse function depending on 75 variables and fixed all the components of the global minimizer except for two of them. Figure 1 shows the plot of the Morse function f depending on x^{55} and x^{56}; it also shows the partitioning scheme. Figure 2 shows the covering parallelepipeds for two partitioning schemes (*a* corresponds to the partitioning with respect to the

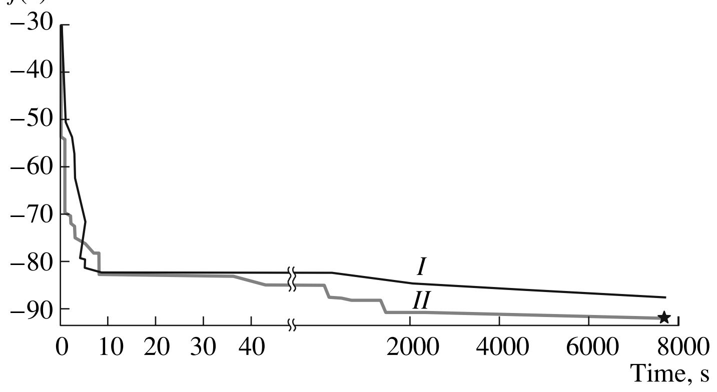

Fig. 3.

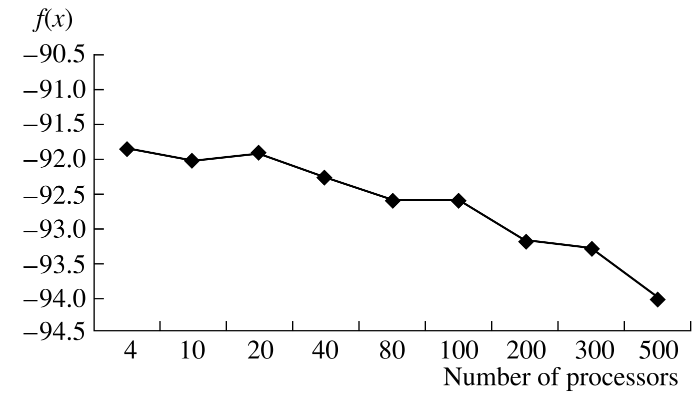

Fig. 4.

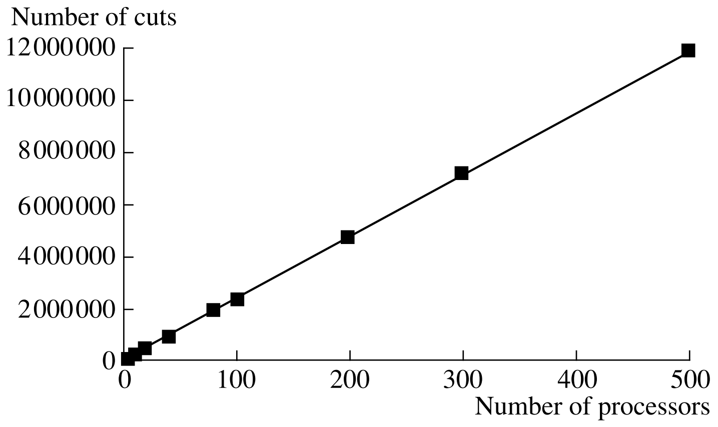

Fig. 5.

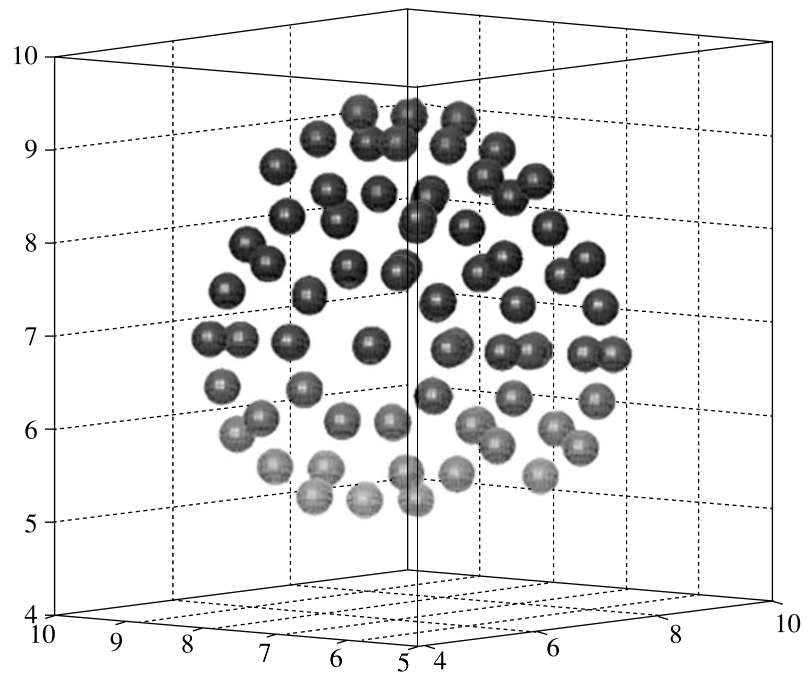

Fig. 6.

longest edge and b with respect to the scalable edge) with respect to the coordinates x^{10} and x^{13}. (The asterisk in Figs. 1 and 2 indicates the global minimizer, and, in Fig. 3, it indicates the global minimum).

The results in Table 1 show that strategy II reduces the number of cuts required to find the minimum by several fold. Moreover, the reliability of the estimates provided by strategy II is higher by an order of magnitude because it does not prune the parallelepipeds that can contain the minimizer, which improves the probability to find it.

Figure 3 shows the comparative plot of finding the minimum of the Morse function with 75 free variables and $\rho = 3$ for two strategies. It is seen from this plot that strategy II yields the minimum much faster.

Table 2 presents the results of the comparison of three methods for selecting the next working parallelepiped—MINQ (according to the minimal estimate), FIFO (first in first out), and LIFO (last in first out).

Figures 4 a 5, respectively, illustrate the dependence of the minimum and of the number of cuts on the number of processors for a fixed value of the computation time. The numerical results show that the minimum is considerably improved as the number of processors increases and the number of cuts is proportional to the number of processors. Figure 6 shows the structure of a cluster with $n = 85$ and $\rho = 6$.

7. CONCLUSIONS

The variants of the method of nonuniform coverings proposed in this paper make it possible to globally optimize Lipschitzian functions. The requirements that the Lipschitz constant is known and invariable in the entire search domain are removed. The differentiability condition of the function is introduced. Due to this condition, the lower bounds of the function values in the current search domains are considerably improved, which reduces the computation time by accelerating the pruning of "unpromising" parallelepipeds. The proposed parallel global optimization method can be used to solve multicriteria optimization problems. The simplest approach is to use the results obtained in [13]. The proposed method can also be used to find the global minimax as in [14]. In addition, it can be used to minimize the Lipschitzian functions under the additional requirement that some of the components of the vector x are integer.

ACKNOWLEDGMENTS

This study was supported by fundamental research program no. 14 of the Presidium of the Russian Academy of Sciences (Part II: High-Performance Computations and Multiprocessor Systems), by program no. 15 of the Presidium of the Russian Academy of Sciences (Study and Creation of Scientific Applications in the Distributed Supercomputer Environment of the Dorodnicyn Computing Center of the Russian Academy of Sciences, 2007), and by grant NSh-5073.2008.1 of the President of the Russian Federation.

We are grateful to A.Ya. Belyankov for useful discussions and valuable remarks.

REFERENCES

1. R. C. Strongin and Ya. D. Sergeyev, *Global Optimization with Non-Convex Constraints. Sequential and Parallel Algorithms* (Kluwer, Dordrecht, 2000).
2. V. V. Voevodin and Vl. V. Voevodin, *Parallel Computations* (BKhV-Peterburg, St. Petersburg, 2002) [in Russian].
3. Yu. G. Evtushenko, "A Numerical Method for Global Optimization of Functions (Search on a Nonuniform Grid)," Zh. Vychisl. Mat. Mat. Fiz. **11**, 1390–1403 (1971).
4. Yu. G. Evtushenko and V. A. Rat'kin, "Bisection Method for the Global Optimization of Functions of Several Variables," Izv. Ross. Akad. Nauk, Tekh. Kibern., No. 1, 119–127 (1987).
5. A. Ya. Belyankov, "Improving the Efficiency of Nonuniform Covering Methods in Global Optimization," in *Abstracts of the Conference on Mathematical Programming and Software* (Ural'skoe Otdelenie Akad. Nauk SSSR, Sverdlovsk, 1989), pp. 21–22 [in Russian].
6. Yu. G. Evtushenko, V. U. Malkova, and A. A. Stanevichyus, "Parallelization of the Global Extremum Searching Process," Avtom.Telemekh., No. 5, 46–58 (2007) [Autom. Remote Control **68**, 787–798 (2007)].
7. Yu. Nesterov and B. Polyak, "Cubic Regularization of Newton Method and Its Global Performance," Math. Program. **108** (1), 177–205 (2006).
8. A. S. Strekalovsky, *Elements of Nonconvex Optimization* (Nauka, Novosibirsk, 2003) [in Russian].

9. A. Ya. Belyankov, “Preliminary Parallelipiped Partitioning in Nonuniform Covering Methods in Global Optimization,” *II All-Russia Conf. with Youth Research School on Mathematical Modeling of Developing Economies* (Vyatskii Gos. Univ., Kirov, 2007), pp. 59–62 [in Russian].
10. M. A. Posypkin and I. Kh. Sigal, “Investigation of Algorithms for Parallel Computations in Knapsack-Type Discrete Optimization Problems,” Zh. Vychisl. Mat. Mat. Fiz. **45** (10), 1801–1809 (2005) [Comput. Math. Math. Phys. **45**, 1735–1742 (2005)].
11. Joint Supercomputer Center of the Russian Academy of Sciences, httr://www.jsss.ru
12. Cambridge Cluster Database, http://www-wales.ch.cam.ac.uk/jon/structures/Morse.html.
13. Yu. G. Evtushenko and M. A. Potapov, “Methods for Solving Multicriteria Problems,” Dokl. Akad. Nauk SSSR **291**, 25–29 (1986).
14. Yu. G. Evtushenko, “A Numerical Method for Finding Best Guaranteed Estimates,” Zh. Vychisl. Mat. Mat. Fiz. **12** (1), 89–104 (1972).