УДК 519.626

ОДНОРОДНЫЕ АЛГОРИТМЫ МНОГОЭКСТРЕМАЛЬНОЙ ОПТИМИЗАЦИИ ДЛЯ ЦЕЛЕВЫХ ФУНКЦИЙ СО ЗНАЧИТЕЛЬНЫМ ВРЕМЕНЕМ ВЫЧИСЛЕНИЯ ЗНАЧЕНИЯ

С. М. Елсаков[1], **В. И. Ширяев**[1]

Предлагаются способы ускорения однородных алгоритмов глобальной оптимизации. Доказаны теоремы о возможностях ускорения без потери сходимости к глобальному минимуму. Рассматриваются модели целевых функций. Доказано, что использование этих моделей обеспечивает сходимость к глобальному минимуму целевой функции. Построена модель для определения области применения алгоритма. Представлены результаты численного тестирования предложенного алгоритма. По результатам тестирования сделан вывод о его области применения.

Ключевые слова: глобальная оптимизация, многоэкстремальная оптимизация, однородные алгоритмы, поверхности отклика, модели целевых функций.

Введение. Многие задачи принятия решений можно сформулировать как задачи оптимизации. Такие задачи возникают в самых разных областях: проектирование систем навигации [2], химико-технологических систем [1] и многих других [16].

Сложности решения этих задач связаны с возрастающей сложностью моделей и, следовательно, самих задач оптимизации. При решении многих задач оптимизации невозможно применять аналитические методы как ввиду сложности целевой функции, так и ввиду алгоритмического задания целевой функции и/или ограничений. Часто такие задачи [17] выделяют в отдельный класс задач с целевыми функциями в виде "черного ящика". Кроме того, в этих задачах существенную роль играет время вычисления целевой функции.

Рассматривается задача многоэкстремальной оптимизации

$$f^* = f\big(x^*\big) = \min_{x\in X} f(x), \quad X \subset \mathbb{R}^d, \quad X = [a;\, b] = \big\{x \in \mathbb{R}^d \ : -\infty < a_j \leqslant x_j \leqslant b_j < \infty,\ j = \overline{1,d}\,\big\}, \qquad (1)$$

где f^* — целевое значение, глобальный минимум, $x^* \in \mathbb{R}^d$ — точка допустимого множества, в которой достигается глобальный минимум, $f(x)$ — целевая функция, X — допустимое множество, d — размерность задачи. Без потери общности можно считать, что $X = \big\{x \in \mathbb{R}^d :\ 0 \leqslant x_j \leqslant 1,\ j = \overline{1,d}\,\big\}$. Предполагается, что функция $f(x)$ удовлетворяет условию Липшица $\big|f(x_1) - f(x_2)\big| \leqslant L\|x_1 - x_2\|, \quad x_1, x_2 \in X$, где L — константа Липшица, а $\|\cdot\|$ обозначает евклидову норму.

Нашей целью будет построение алгоритма глобальной оптимизации, решающего задачу (1). Настоящая статья продолжает исследования, начатые авторами в [6, 14]. Статья организована следующим образом. В разделе 1 приводятся основные необходимые результаты из [6]. В разделе 2 предлагаются способы ускорения однородных алгоритмов. В разделе 3 предлагаются модели целевых функций и показывается, что эти модели позволяют построить алгоритмы глобальной оптимизации, сходящиеся к глобальному минимуму целевой липшицевой функции. В разделе 4 приводятся результаты сравнения предложенных алгоритмов с различными классами алгоритмов и обсуждается область применимости одного из предлагаемых алгоритмов.

1. Класс однородных алгоритмов глобальной оптимизации. Пусть $\big\{x_i\big\}_{i=1}^k$ — последовательность точек допустимого множества, в которых некоторым алгоритмом оптимизации осуществляется вычисление значений целевой функции. Будем говорить, что алгоритм относится к классу однородных алгоритмов глобальной оптимизации, если одновременно выполнены следующие три условия.

1. Алгоритм является *однородным*, т.е. для любых двух целевых функций, различающихся на константу, последовательности точек испытаний $\big\{x_i\big\}_{i=1}^k$ совпадают.

2. Функции $m_k(x) = m\big(x, x_1, f(x_1), \ldots, x_k, f(x_k)\big)$ и $s_k(x) = s\big(x, x_1, f(x_1), \ldots, x_k, f(x_k)\big)$ определены на каждой итерации алгоритма и удовлетворяют условиям

[1] Южно-Уральский государственный университет, механико-математический факультет, просп. Ленина, 87, 454080, г. Челябинск; С. М. Елсаков, математик, e-mail: esergeym@mail.ru; В. И. Ширяев, зав. кафедрой, профессор, e-mail: vis@prima.susu.ac.ru

A1) $m_k(x_i) = f(x_i), \quad i = \overline{1,k}$;
A2) $s_k(x_i) = 0, \quad i = \overline{1,k}$;
A3) $s_k(x) > 0, \quad x \neq x_i, \quad i = \overline{1,k}$;
A4) $m_k(x)$, $s_k(x)$ — липшицевы функции.
3. Алгоритм может быть представлен в виде следующих шагов.

Шаг 1. Выбрать начальные точки $x_i \in \mathbb{R}^d$, $i = \overline{1,M}$. Положить $k = M$.

Шаг 2. Вычислить

$$x_{k+1} = \operatorname*{argmin}_{x \in X} P\big(m_k(x), s_k(x)\big), \tag{2}$$

где $P\big(m_k(x), s_k(x)\big)\colon R^2 \to R$ — функция выбора, которая единственна для алгоритма, $m_k(x)$, $s_k(x)$ — функции, которые перестраиваются после каждой итерации алгоритма и удовлетворяют условиям A1–A4.

Шаг 3. Вычислить значение $f(x_{k+1})$, положить $k = k + 1$.

Шаг 4. Если выполнено условие выхода φ, то выйти, иначе вернуться на шаг 2.

Многие существующие алгоритмы многоэкстремальной оптимизации являются однородными: информационно-статистический алгоритм Р. Г. Стронгина [11], алгоритм ломаных С. А. Пиявского [8] и многие другие [5]. Отметим, что, несмотря на отсутствие учета дополнительной информации, например градиентов, в алгоритме, эта информация может быть учтена при построении функций $m_k(x)$ и $s_k(x)$.

Для того чтобы реализовать численный алгоритм, требуется в указанном однородном алгоритме конкретизировать функцию $P(\cdot,\cdot)$, алгоритм построения функций $m_k(x)$ и $s_k(x)$, алгоритм решения задачи оптимизации (2) и условие остановки φ.

1.1. Структура и условия сходимости однородных алгоритмов глобальной оптимизации. Выясним вид функции $P(\cdot,\cdot)$, позволяющей использовать для построения однородного алгоритма различные виды функций $m_k(x)$ и $s_k(x)$.

Теорема 1. *Если существует такая дважды непрерывно дифференцируемая функция $P(m,s)$, что*
1) *для любых функций $m_k(x)$ и $s_k(x)$, удовлетворяющих условиям* A1–A4 *и условиям*
A5) $m\big(x, x_1, f(x_1) + c, \ldots, x_k, f(x_k) + c\big) = m\big(x, x_1, f(x_1), \ldots, x_k, f(x_k)\big) + c, \quad k \in \mathbb{N}, \quad c \in \mathbb{R}$;
A6) $s\big(x, x_1, f(x_1) + c, \ldots, x_k, f(x_k) + c\big) = s\big(x, x_1, f(x_1), \ldots, x_k, f(x_k)\big), \ k \in N, \ c \in \mathbb{R}$, *выполняется условие* $\operatorname*{argmin}_{x \in X} P\big(m_k(x), s_k(x)\big) = \operatorname*{argmin}_{x \in X} P\big(m_k(x) + c, s_k(x)\big)$,
2) *для любой точки $\big(m^*, s^*\big)$ существуют функции $m_k^*(x)$ и $s_k^*(x)$, удовлетворяющие условиям* A1–A6, *такие, что решение задачи* (2) *достигается в точке $\big(m^*, s^*\big)$,*
то тогда для $P(m,s)$ выполняется равенство $\operatorname*{argmin}_{x \in X} P\big(m_k(x), s_k(x)\big) = \operatorname*{argmin}_{x \in X}\Big[\overline{C} m_k(x) + p\big(s_k(x)\big)\Big]$,
где $\overline{C} = \text{const}$ и $p(s)$ — дважды дифференцируемая функция.

Доказательство этого утверждения приведено в [6]. Для того чтобы можно было выбирать функции $m_k(x)$ и $s_k(x)$ независимо, сформулируем условия, которые бы обеспечивали сходимость алгоритма к точкам глобального минимума.

Теорема 2. *Для того чтобы множество предельных точек последовательности $\big\{x_i\big\}_{i=1}^{\infty}$, порождаемых однородным алгоритмом с критерием $P\big(m_k(x), s_k(x)\big) = Cm_k(x) + p\big(s_k(x)\big)$, совпадало с множеством глобальных минимумов липшицевой функции $f(x)$ с константой Липшица L на компакте X, достаточно, чтобы функция $p(\cdot)$ была липшицевой и при $K = \text{const}$ и $K > L$ выполнялось неравенство*

$$\min_{x \in X}\Big(P\big(m_k(x), s_k(x)\big)\Big) \leqslant \min_{x \in X} \max_{i=\overline{1,k}} \big(f(x_i) - K\|x - x_i\|\big). \tag{3}$$

Доказательство этого утверждения приведено в [6]. Условие (3) является удобным в теоретическом плане, однако для построения алгоритма удобнее было бы сформулировать его в терминах функций $m_k(x)$ и $s_k(x)$.

Теорема 3. *Если для функции $s_k(x)$ выполняется условие*
A7) $s_k(x) \geqslant \min\limits_{i=\overline{1,k}} \|x - x_i\|$,

то множество предельных точек последовательности $\{x_i\}_{i=1}^{\infty}$, *порождаемых алгоритмом с функцией* $P\big(m_k(x), s_k(x)\big) = m_k(x) - 2Ks_k(x)$, *где* $K > L + L_m$ *и* L_m — *константа Липшица для функции* $m_k(x)$, *будет совпадать с множеством глобальных минимумов липшицевой функции* $f(x)$ *с константой Липшица* L.

Доказательство этого утверждения приведено в [6]. Для определения однородного алгоритма требуется предложить правило остановки φ.

Теорема 4. *Если в качестве критерия остановки* φ *выбрать условие вида* $\min\limits_{i=\overline{1,k}} \|x_i - x_{k+1}\| < \varepsilon$, *то однородный алгоритм обеспечит решение задачи* (1) *с точностью по значению целевой функции не хуже, чем* $L\,\dfrac{L_p + L_m}{K - L}\,\varepsilon$, *где* L_p — *константа Липшица для функции* $p\big(s_k(x)\big)$ *по аргументу* x.

Доказательство этого утверждения приведено в [6]. Эти теоремы позволяют свести задачу построения алгоритма глобальной оптимизации к задаче выбора функций $m_k(x)$ и $s_k(x)$.

2. Способы ускорения однородных алгоритмов без потери сходимости.

2.1. Ускорение решения вспомогательной задачи. Предварительное тестирование алгоритмов продемонстрировало, что решение на каждом шаге задачи (2) как задачи глобальной оптимизации является чрезвычайно трудоемкой операцией и для построения практического алгоритма требуется отказаться от применения глобальной оптимизации на каждом шаге. Одна из возможностей снизить объем вспомогательных вычислений — решать задачу (2) не точно, а с некоторой погрешностью, поскольку, с одной стороны, находить глобальный минимум на каждой итерации может оказаться слишком затратным, а с другой — точность решения вспомогательной задачи должна быть согласована с требуемой точностью решения задачи (1).

Теорема 5. *Если на* k-*м шаге не удалось найти такую точку* x_{k+1}, *чтобы выполнялось условие*

$$P_k(x_{k+1}) = P\big(m_k(x_{k+1}), s_k(x_{k+1})\big) \leqslant \min_{i=\overline{1,k}} f(x_i) - \varepsilon, \tag{4}$$

то задача глобальной оптимизации (1) *решена с точностью* ε *по значению целевой функции.*

Доказательство. Пусть на очередном шаге не удалось найти точку x_{k+1}, такую, чтобы выполнялось условие (4). Это означает, что $\min\limits_{i=\overline{1,k}} f(x_i) - \varepsilon < \min\limits_{x\in X} P_k(x)$. Учтем, что выполнены условия A1–A7, а значит [6] выполняется условие (3): $\min\limits_{x\in X}\Big(P\big(m_k(x), s_k(x)\big)\Big) \leqslant \min\limits_{x\in X}\max\limits_{i=\overline{1,k}}\big(f(x_i) - K\|x - x_i\|\big)$.

Выпишем совместно два эти неравенства:

$$\min_{i=\overline{1,k}} f(x_i) - \varepsilon < \min_{x\in X} P\big(m_k(x), s_k(x)\big) \leqslant \min_{x\in X}\max_{i=\overline{1,k}}\big(f(x_i) - K\|x - x_i\|\big),$$
$$\min_{i=\overline{1,k}} f(x_i) - \varepsilon < \min_{x\in X}\max_{i=\overline{1,k}}\big(f(x_i) - K\|x - x_i\|\big) \leqslant \min_{x\in X}\max_{i=\overline{1,k}}\big(f(x_i) - L\|x - x_i\|\big).$$

Последнее неравенство и означает, что задача решена с точностью ε по значению целевой функции, что и требовалось доказать.

Таким образом, на каждом шаге достаточно получать лишь такую точку x_{k+1}, чтобы она удовлетворяла условию (4). Если же такая точка не может быть найдена, то либо задача решена с требуемой точностью, либо использованная оценка константы Липшица является заниженной. Следовательно, выполнение неравенства (4) может быть использовано как индикатор необходимости увеличить оценку константы Липшица.

Соответственно, может быть предложен алгоритм для решения вспомогательной задачи.

Шаг 0 (выполняется при инициализации однородного алгоритма). Выбрать значение коэффициента достоверности $r_0 = 1.0$.

Шаг 1. Если $\operatorname{argmin}\limits_{i=\overline{1,k}} f(x_i) \geqslant k + 1 - 2^{d+1}$, то положить $r_1 = 0$, иначе перейти на шаг 6.

Шаг 2. Положить $K = r_1 \max\limits_{\substack{i,j=\overline{1,k}\\ i\neq j}} \dfrac{\big|f(x_j) - f(x_i)\big|}{\|x_j - x_i\|}$. Запустить локальный поиск из точки $\operatorname{argmin}\limits_{x_i,\, i=\overline{1,k}} f(x_i)$ с константой Липшица $1.3\big(\widehat{L}_m + \widehat{L}\widehat{L}_s\big)$. Найденную точку обозначить $\widehat{x}_{k+1}$.

Шаг 3. Если точка $\widehat{x}_{k+1}$ удовлетворяет условию

$$\Big(r_1 = 0 \wedge \min_{i=\overline{1,k}} \big\|x_i - \widehat{x}_{k+1}\big\| \leqslant 0.5T\Big) \vee \Big(r_1 \neq 0 \wedge P\big(\widehat{x}_{k+1}\big) \geqslant \min_{i=\overline{1,k}} f(x_i) - r_1\widehat{L}T\Big),$$

то если $r_1 \geqslant 3$, то перейти на шаг 4, иначе положить $r_1 = r_1 + 0.1$ и перейти на шаг 2, в противном случае перейти на шаг 16.

Шаг 4. Положить $K = r_0 \max\limits_{\substack{i,j=\overline{1,k}\\ i\neq j}} \dfrac{\big|f(x_j) - f(x_i)\big|}{\|x_j - x_i\|}$. Если

$$\max\left\{ i,\ i = \overline{2,k} \,\middle|\, \frac{\big|f(x_i) - m_{i-1}(x_i)\big|}{s_{i-1}(x_i) \max\limits_{1\leqslant j\neq t\leqslant i-1} \dfrac{\big|f(x_j) - f(x_t)\big|}{\|x_j - x_t\|}} \leqslant -0.25 \right\} \geqslant k - 1 + 2^{d+1},$$

то запустить локальный поиск из точки x с индексом

$$\max\left\{ i,\ i = \overline{2,k} \,\middle|\, \frac{\big|f(x_i) - m_{i-1}(x_i)\big|}{s_{i-1}(x_i) \max\limits_{1\leqslant j\neq t\leqslant i-1} \dfrac{\big|f(x_j) - f(x_t)\big|}{\|x_j - x_t\|}} \leqslant -0.25 \right\}$$

с константой Липшица $1.3\big(\widehat{L}_m + \widehat{L}\widehat{L}_s\big)$. Найденную точку обозначить $\widehat{x}_{k+1}$.

Шаг 5. Если точка $\widehat{x}_{k+1}$ удовлетворяет условию $P\big(\widehat{x}_{k+1}\big) \geqslant \min\limits_{i=\overline{1,k}} f(x_i) - r_0\widehat{L}T$, то перейти на шаг 16.

Шаг 6. Запустить локальный поиск из точки $\operatorname{argmin}\limits_{x_i,\, i=\overline{1,k}} f(x_i)$ с константой Липшица $1.3\big(\widehat{L}_m + \widehat{L}\widehat{L}_s\big)$. Найденную точку обозначить $\widehat{x}_{k+1}$.

Шаг 7. Если точка $\widehat{x}_{k+1}$ удовлетворяет условию $P\big(\widehat{x}_{k+1}\big) \geqslant \min\limits_{i=\overline{1,k}} f(x_i) - r_0\widehat{L}T$, то перейти на шаг 16.

Шаг 8. Запустить локальный поиск из точки x_k с константой Липшица $1.3\big(\widehat{L}_m + \widehat{L}\widehat{L}_s\big)$. Найденную точку обозначить $\widehat{x}_{k+1}$.

Шаг 9. Если точка $\widehat{x}_{k+1}$ удовлетворяет условию $P\big(\widehat{x}_{k+1}\big) \geqslant \min\limits_{i=\overline{1,k}} f(x_i) - r_0\widehat{L}T$, то перейти на шаг 16.

Шаг 10. Для всех точек y_j, $j = \overline{1,t}$ выполнить локальный поиск из точки y_j с константой Липшица $1.3\big(\widehat{L}_m + \widehat{L}\widehat{L}_s\big)$. Найденную точку обозначить $\widehat{x}_{k+1}$.

Шаг 11. Если точка $\widehat{x}_{k+1}$ удовлетворяет условию $P\big(\widehat{x}_{k+1}\big) \geqslant \min\limits_{i=\overline{1,k}} f(x_i) - r_0\widehat{L}T$, то перейти на шаг 16, иначе, если множество точек исчерпано, то перейти на шаг 12, в противном случае выбрать следующую точку y_j и перейти на шаг 10.

Шаг 12. Пусть $i = \operatorname{argmin}\limits_{i=\overline{1,k}} \Big[f(x_i) + f(x_{j(i)}) - 1.3\big(\widehat{L}_m + \widehat{L}\widehat{L}_s\big)\|x_i - x_{j(i)}\|\Big]$, где $j(i) = \operatorname{argmin}\limits_{j=\overline{1,k}} \|x_j - x_i\|$; тогда запустить локальный поиск из точки $\alpha x_i + (1-\alpha)x_{j(i)}$, где $\alpha = 0.5 + 0.5\,\dfrac{f(x_i) - f(x_{j(i)})}{1.3\big(\widehat{L}_m + \widehat{L}\widehat{L}_s\big)}$ с константой Липшица $1.3\big(\widehat{L}_m + \widehat{L}\widehat{L}_s\big)$. Найденную точку обозначить $\widehat{x}_{k+1}$.

Шаг 13. Если точка $\widehat{x}_{k+1}$ удовлетворяет условию $P\big(\widehat{x}_{k+1}\big) \geqslant \min\limits_{i=\overline{1,k}} f(x_i) - r_0\widehat{L}T$, то перейти на шаг 16.

Шаг 14. Выполнить глобальный поиск с константой Липшица $1.2\widehat{L}$. Из найденной точки выполнить локальный поиск с константой Липшица $1.3\big(\widehat{L}_m + \widehat{L}\widehat{L}_s\big)$. Найденную точку обозначить $\widehat{x}_{k+1}$.

Шаг 15. Если точка $\widehat{x}_{k+1}$ удовлетворяет условию $P\big(\widehat{x}_{k+1}\big) \geqslant \min\limits_{i=\overline{1,k}} f(x_i) - r_0\widehat{L}T$, то перейти на шаг 16, иначе положить $r_0 = r_0 + 0.1$ и перейти на шаг 4.

Шаг 16. Положить $x_{k+1} = \widehat{x}_{k+1}$.

В описании алгоритма использованы следующие обозначения: r_0 — коэффициент достоверности, d — размерность пространства, r_1 — коэффициент достоверности, который подбирается в том случае, если алгоритм нашел значение лучше всех предыдущих, $\widehat{L}$ — оценка константы Липшица для функции $f(x)$, $\widehat{L}_m$ — оценка константы Липшица для функции $m_k(x)$, $\widehat{L}_s$ — оценка константы Липшица для функции $s_k(x)$, T — требуемая точность решения задачи (1) по аргументу, y_j — точки допустимого множества, полученные при последнем выполнении процедуры глобальной оптимизации и удовлетворяющие условию $P(y_j) \geqslant \min\limits_{i=\overline{1,k}} f(x_i) - r_0\widehat{L}T$.

Отметим некоторые особенности представленного алгоритма. Прежде всего, в этом алгоритме одновременно решаются две задачи: ищется минимум функции $P(x)$ и подбирается значение для доверительного коэффициента в оценки константы Липшица. На всех шагах, кроме шага 14, для работы используется алгоритм локальной оптимизации, причем задача одномерной оптимизации решается с помощью алгоритма ломаных [8]. На шаге 14 применяется алгоритм глобальной оптимизации [7] с заданной оценкой константы Липшица, при этом любая точка, удовлетворяющая неравенству (4), принимается не сразу, а спустя 1000 итераций алгоритма для того, чтобы была возможность сформировать непустое множество точек y_j.

2.2. Ускорение построения функций $m_k(x)$ и $s_k(x)$. Построение алгоритма глобальной оптимизации без использования разбиения допустимого множества может приводить к значительным вычислительным затратам при большом количестве испытаний целевой функции. Например, если в качестве функции $m_k(x)$ использовать RBF-сплайн, то с ростом количества испытаний k построение функции $m_k(x)$ будет требовать обращения матрицы размером порядка $k \times k$. Для того чтобы снизить требуемое время, рассмотрим возможность использования разбиения допустимого множества на подмножества X_i, $i = \overline{1,M}$. Для этого все допустимое множество разбивается на пересекающиеся подмножества [28], внутри каждого строится необходимая функция по испытаниям, принадлежащим только этому подмножеству, а в точках, где пересекаются несколько подмножеств, используется линейная свертка этих вспомогательных функций со специально подобранными коэффициентами.

Пусть допустимое множество X разбито на подмножества $X = X_1 \cup X_2 \cup \ldots \cup X_M$ таким образом, что $X_i \cap X_j \neq \emptyset$, $i, j = \overline{1,M}$, и количество точек испытаний внутри каждого подмножества отлично от нуля. Обозначим границу множества X через ∂X, а множеств X_i через ∂X_i. Введем расстояние до внутренней границы подмножества как $\rho_i(x) = \min\limits_{y \in \partial X_i \setminus \partial X} \|x - y\|$.

Теорема 6. *Пусть существует алгоритм построения функций $m_k(x)$ и $s_k(x)$, удовлетворяющих условиям* A1–A7, *и разбиение таково, что для любой точки $x \in X$, которая принадлежит более чем одному подмножеству X_i, выполняется неравенство $\sum\limits_{i:\, x \in X_i} \rho_i(x) > \delta > 0$, тогда функции*

$$m_k(x)' = \left(\sum_{x \in X_i} \rho_i(x) m_k^i(x)\right)\left(\sum_{x \in X_i} \rho_i(x)\right)^{-1}, \quad s_k(x)' = \left(\sum_{x \in X_i} \rho_i(x) s_k^i(x)\right)\left(\sum_{x \in X_i} \rho_i(x)\right)^{-1},$$

где $m_k^i(x)$, $s_k^i(x)$ — функции $m_k(x)$ и $s_k(x)$, построенные для испытаний, которые располагаются только внутри подмножества X_i, также удовлетворяют условиям

A1) $m_k(x_i) = f(x_i), \quad i = \overline{1,k}$;
A2) $s_k(x_i) = 0, \quad i = \overline{1,k}$;
A3) $s_k(x) > 0, \quad x \neq x_i, \quad i = \overline{1,k}$;
A4) $s_k(x)$ и $m_k(x)$ — *липшицевы функции*;
A5) $m\big(x, x_1, f(x_1) + c, \ldots, x_k, f(x_k) + c\big) = m\big(x, x_1, f(x_1), \ldots, x_k, f(x_k)\big) + c, \quad k \in \mathbb{N}, \quad c \in \mathbb{R}$;
A6) $s\big(x, x_1, f(x_1) + c, \ldots, x_k, f(x_k) + c\big) = s\big(x, x_1, f(x_1), \ldots, x_k, f(x_k)\big), \quad k \in \mathbb{N}, \quad c \in \mathbb{R}$;
A7) $s_k(x) \geqslant \min\limits_{i=\overline{1,k}} \|x - x_i\|$.

Доказательство. Рассмотрим выполнение условий A1–A7 для функций $m_k(x)'$ и $s_k(x)'$. Условия A1–A3 выполняются по построению функций $m_k(x)'$ и $s_k(x)'$. Проверим выполнение условия A4.

Пусть точки x_1 и x_2 принадлежат одинаковому набору подмножеств X_i, $i = \overline{1,M}$. Рассмотрим функцию $\rho_i(x)$ и покажем, что она является липшицевой. Пусть $\rho_i(x_1) = \|x_1 - y_1\|$, $y_1 \in X_i$, $\rho_i(x_2) = \|x_2 - y_2\|$,

$y_2 \in X_i$, и пусть для определенности $\|x_1 - y_1\| \geqslant \|x_2 - y_2\|$, тогда с учетом неравенства треугольника имеем

$$\big|\rho_i(x_1) - \rho_i(x_2)\big| = \|x_1 - y_1\| - \|x_2 - y_2\| \leqslant \|x_1 - y_2\| - \|x_2 - y_2\| \leqslant \|x_1 - x_2\|.$$

Покажем липшицевость функции $\rho_j(x)\left(\sum\limits_{i=1}^{M}\rho_i(x)\right)^{-1}$:

$$\begin{aligned}
\left|\frac{\rho_j(x_1)}{\sum\limits_{i=1}^{M}\rho_i(x_1)} - \frac{\rho_j(x_2)}{\sum\limits_{i=1}^{M}\rho_i(x_2)}\right| &= \left|\frac{\rho_j(x_1)\sum\limits_{i=1}^{M}\rho_i(x_2) - \rho_j(x_2)\sum\limits_{i=1}^{M}\rho_i(x_1)}{\sum\limits_{i=1}^{M}\rho_i(x_1)\sum\limits_{i=1}^{M}\rho_i(x_2)}\right| = \\
&= \left|\frac{\rho_j(x_1)\sum\limits_{i=1}^{M}\rho_i(x_2) - \rho_j(x_1)\sum\limits_{i=1}^{M}\rho_i(x_1) + \rho_j(x_1)\sum\limits_{i=1}^{M}\rho_i(x_1) - \rho_j(x_2)\sum\limits_{i=1}^{M}\rho_i(x_1)}{\sum\limits_{i=1}^{M}\rho_i(x_1)\sum\limits_{i=1}^{M}\rho_i(x_2)}\right| = \\
&= \left|\frac{\rho_j(x_1)\sum\limits_{i=1}^{M}\big[\rho_i(x_2) - \rho_i(x_1)\big] + \big[\rho_j(x_1) - \rho_j(x_2)\big]\sum\limits_{i=1}^{M}\rho_i(x_1)}{\sum\limits_{i=1}^{M}\rho_i(x_1)\sum\limits_{i=1}^{M}\rho_i(x_2)}\right| \leqslant \\
&\leqslant \frac{M\max\big|\rho_i(x_1)\big| + \max\left|\sum\limits_{i=1}^{M}\rho_i(x_1)\right|}{\min\left|\sum\limits_{i=1}^{M}\rho_i(x_1)\sum\limits_{i=1}^{M}\rho_i(x_2)\right|}\,\|x_1 - x_2\| \leqslant \frac{2M\max\big|\rho_i(x_1)\big|}{\delta^2}\,\|x_1 - x_2\|.
\end{aligned}$$

Теперь рассмотрим липшицевость функции $m_k(x)'$:

$$\begin{aligned}
\big|m_k(x_1)' - m_k(x_2)'\big| &= \left|\frac{\sum\limits_{i=1}^{M}\rho_i(x_1)m_k^i(x_1)}{\sum\limits_{i=1}^{M}\rho_i(x_1)} - \frac{\sum\limits_{i=1}^{M}\rho_i(x_2)m_k^i(x_2)}{\sum\limits_{i=1}^{M}\rho_i(x_2)}\right| = \\
&= \left|\frac{\sum\limits_{i=1}^{M}\rho_i(x_1)m_k^i(x_1)}{\sum\limits_{i=1}^{M}\rho_i(x_1)} - \frac{\sum\limits_{i=1}^{M}\rho_i(x_2)m_k^i(x_1)}{\sum\limits_{i=1}^{M}\rho_i(x_2)} + \frac{\sum\limits_{i=1}^{M}\rho_i(x_2)m_k^i(x_1)}{\sum\limits_{i=1}^{M}\rho_i(x_2)} - \frac{\sum\limits_{i=1}^{M}\rho_i(x_2)m_k^i(x_2)}{\sum\limits_{i=1}^{M}\rho_i(x_2)}\right| = \\
&= \left|\sum_{i=1}^{M}m_k^i(x_1)\left[\frac{\rho_i(x_1)}{\sum\limits_{j=1}^{M}\rho_j(x_1)} - \frac{\rho_i(x_2)}{\sum\limits_{j=1}^{M}\rho_j(x_2)}\right] + \frac{\sum\limits_{i=1}^{M}\rho_i(x_2)\big[m_k^i(x_1) - m_k^i(x_2)\big]}{\sum\limits_{i=1}^{M}\rho_i(x_2)}\right| \leqslant \\
&\leqslant \sum_{i=1}^{M}\big|m_k^i(x_1)\big|\,\frac{2M\max\big|\rho_i(x_1)\big|}{\delta^2}\,\|x_1 - x_2\| + L_m\|x_1 - x_2\| \leqslant \\
&\leqslant \left(\frac{2M^2\max\big|m_k^i(x_1)\big|\max\big|\rho_i(x_1)\big|}{\delta^2} + L_m\right)\|x_1 - x_2\|.
\end{aligned}$$

Учтем, что функция $m_k^i(x_1)$ удовлетворяет условию A4, а значит существует L_m — константа Липшица функции $m_k^i(x_1)$. С другой стороны, поскольку допустимое множество является компактом, то $\operatorname{diam}(X) < \infty$. Воспользуемся этим и запишем

$$\begin{aligned}
&\left(\frac{2M^2\max\big|m_k^i(x_1)\big|\max\big|\rho_i(x_1)\big|}{\delta^2} + L_m\right)\|x_1 - x_2\| \leqslant \\
&\qquad\leqslant \left(\frac{2M^2\Big[\min\limits_{x_i\in X_i}\big|f(x_i)\big| + \operatorname{diam}(X)L_m\Big]\operatorname{diam}(X)}{\delta^2} + L_m\right)\|x_1 - x_2\|.
\end{aligned}$$

Таким образом, все допустимое множество разбито на подмножества. Внутри каждого из подмножеств функция $m_k(x)'$ липшицева, следовательно, она липшицева и на всем допустимом множестве. Аналогично можно показать липшицевость функции $s_k(x)'$. Проверим выполнение условия A5:

$$m_k\big(x, x_1, f(x_1)+c, \ldots, x_k, f(x_k)+c\big)' = \frac{\sum\limits_{x\in X_i} \rho_i(x) m_k^i\big(x, x_1, f(x_1)+c, \ldots, x_k, f(x_k)+c\big)}{\sum\limits_{x\in X_i} \rho_i(x)} =$$

$$= \frac{\sum\limits_{x\in X_i} \rho_i(x)\Big[m_k^i\big(x, x_1, \ldots, f(x_k)\big)+c\Big]}{\sum\limits_{x\in X_i} \rho_i(x)} =$$

$$= m_k\big(x, x_1, f(x_1), \ldots, x_k, f(x_k)\big)' + c.$$

Аналогично проверяется условие A6. Осталось проверить условие A7:

$$s_k(x)' = \frac{\sum\limits_{x\in X_i} \rho_i(x) s_k^i(x)}{\sum\limits_{x\in X_i} \rho_i(x)} \geqslant \frac{\sum\limits_{x\in X_i} \rho_i(x) \min\limits_{x_i\in X_i} \|x-x_i\|}{\sum\limits_{x\in X_i} \rho_i(x)} = \min_{x_i\in X_i} \|x-x_i\|.$$

Теорема 6 доказана.

С учетом доказанной теоремы предлагается следующий алгоритм построения декомпозиции функций $m_k(x)$ и $s_k(x)$ в предположении, что существует алгоритм построения функций $m_k(x)$ и $s_k(x)$, но при значительном количестве точек $\{x_i\}_{i=1}^k$ время работы с функциями $m_k(x)$ и $s_k(x)$ становится неприемлемо большим. Выделим две операции, которые необходимо выполнять с помощью алгоритма: операция добавления точки x_{k+1} к уже имеющимся функциям $m_k(x)$ и $s_k(x)$, т.е. построение функций $m_{k+1}(x)$ и $s_{k+1}(x)$, и операция вычисления значения в заданной точке. Эти операции можно выполнять достаточно эффективно при древовидной декомпозиции исходного допустимого множества — гиперинтервала. Дерево разбиений является бинарным, и каждые два брата являются частями своего родителя, от которого они отсечены гиперплоскостью, ортогональной некоторому ребру.

Введем следующие обозначения: x — точка, в которой вычисляется значение; X_i, $i=\overline{1,N}$, — подмножества допустимого множества (гиперинтервал); S_i, $i=\overline{1,N}$, — множество точек испытания, принадлежащие подмножеству X_i; $m_k^i(x)$ — функция, построенная по точкам из множества S_i. Первоначально полагаем $X_1 = X$, $N=1$, $S_i=\{1\}$. Алгоритм добавления следующей точки имеет такой вид.

Шаг 1. Выбрать все подмножества декомпозиции X_i, которым принадлежит точка x. Для каждого подмножества X_i выполнить следующие шаги.

Шаг 2. Если $\big|S_i\big|+1 < N_{\text{crit}}$ (N_{crit} — параметр алгоритма), то добавить точку в множество S_i, обновить $m_k^i(x)$. **STOP.**

Шаг 3. Если $\big|S_i\big|+1 = N_{\text{crit}}$, то добавить точку x в множество S_i, найти бóльшую сторону j гиперинтервала X_i, упорядочить множество S_i согласно j-координате по возрастанию: $x_{i_1} \leqslant x_{i_2} \leqslant \ldots \leqslant x_{i_{|S_1|}}$.

Шаг 4. Вычислить $b_1 = \Big\lfloor \big|S_i\big|/2 \Big\rfloor + N_{\text{inter}}$ и $b_2 = \Big\lfloor \big|S_i\big|/2 \Big\rfloor - N_{\text{inter}}$ (N_{inter} — параметр алгоритма).

Шаг 5. Если $x_{i_{b_1}} = x_{i_{b_1+1}}$ и $b_1 + 1 < \big|S_i\big|$, то положить $b_1{=}b_1+1$ и повторить шаг 5, в противном случае перейти на Шаг 6.

Шаг 6. Если $x_{i_{b_2}} = x_{i_{b_2-1}}$ и $b_2 > 2$, то положить $b_2 = b_2 - 1$ и повторить шаг 6, иначе перейти на шаг 7.

Шаг 7. Сформировать множества $X_i^1 = \big\{x \in X_i \,|\, x^j \leqslant x^j_{i_{b_1}}\big\}$ и $X_i^2 = \big\{x \in X_i \,|\, x^j \geqslant x^j_{i_{b_2}}\big\}$, соответствующие множествам S_i^1 и S_i^2, и построить соответствующие функции $m_k^i(x)$. **STOP.**

При небольшом количестве испытаний все испытания принадлежат одному подмножеству и декомпозиция не выполняется. Затем при достижении количества испытаний величины N_{crit} производится разделение гиперинтервала по большей диагонали на два пересекающихся гиперинтервала таким образом, чтобы в пересечении новых гиперинтервалов находилось не менее $2N_{\text{inter}}$ точек испытаний. С увеличением количества испытаний дробление гиперинтервалов продолжается.

2.3. Модель для определения области применения алгоритма глобальной оптимизации с существенными вспомогательными вычислениями. По сравнению с другими алгоритмами, например [16, 17, 22], предлагаемые алгоритмы требуют меньшего количества обращений к целевой функции для поиска глобального минимума, но при этом тратят значительно большее время на вспомогательные вычисления. Поэтому рассмотрим модель алгоритма глобальной оптимизации с учетом существенных вспомогательных вычислений.

Будем считать, что алгоритм состоит из двух последовательных шагов, повторяемых многократно. На первом шаге выполняются вспомогательные вычисления, на втором выполняется вычисление целевой функции. Будем полагать, что алгоритм $\mathcal{B}$ — алгоритм с существенными вспомогательными вычислениями, а алгоритм $\mathcal{A}$ — алгоритм с несущественными вспомогательными вычислениями. Введем обозначения: $N_{\mathcal{A}}$ — количество обращений к целевой функции алгоритмом $\mathcal{A}$, $N_{\mathcal{B}}$ — количество обращений к целевой функции алгоритмом $\mathcal{B}$, T_t — время, необходимое для однократного вычисления целевой функции, а T_s — время, затрачиваемое на вспомогательные вычисления алгоритмом $\mathcal{B}$. Время, затрачиваемое на вспомогательные вычисления алгоритмом $\mathcal{A}$, будем полагать нулевым, тогда $N_{\mathcal{A}}T_t$ — время работы алгоритма $\mathcal{A}$, а $N_{\mathcal{B}}T_t + T_s$ — время работы алгоритма $\mathcal{B}$. При условии, что количество обращений к целевой функции алгоритмом $\mathcal{B}$ меньше, чем алгоритмом $\mathcal{A}$, условие применимости алгоритма $\mathcal{B}$ может быть найдено из неравенства $N_{\mathcal{B}}T_t + T_s \leqslant N_{\mathcal{A}}T_t$.

Выразим время вычисления целевой функции

$$T_t \geqslant \frac{T_s}{N_{\mathcal{A}} - N_{\mathcal{B}}}. \tag{5}$$

Таким образом, если предлагаемый алгоритм требует меньшего количества обращений к целевой функции, то при достаточно большом времени вычисления целевой функции предлагаемый алгоритм будет решать задачу быстрее, чем сравниваемый алгоритм.

3. Модели целевых функций для однородных алгоритмов глобальной оптимизации. Для построения однородного алгоритма глобальной оптимизации осталось выбрать конкретный вид функций $m_k(x)$ и $s_k(x)$. Для выбора функций рассмотрим некоторые возможные функции.

3.1. Варианты функции для $m_k(x)$.

3.1.1. Кусочно-линейная функция на основе триангуляции Делоне. Для использования кусочно-линейной функции на основе триангуляции Делоне [9] требуется, чтобы допустимое множество X содержалось в выпуклой оболочке точек $\{x_i\}_{i=1}^k$. Функция $m_k(x)$ задается в виде

$$m_k(x) = \sum_{i=1}^{d+1} a_{ij(x)} f(x_{ij(x)}), \tag{6}$$

где d — размерность пространства, в котором лежит допустимое множество, $j(x)$ обозначает номер симплекса в триангуляции Делоне, которому принадлежит точка x, x_{ij} — вершины симплекса с номером j, a_{ij} — матрица коэффициентов интерполяции, удовлетворяющих условию

$$m_k(x_p) = \sum_{i=1}^{d+1} a_{ij} f(x_{ij}) = f(x_p), \quad p = \overline{1,k}. \tag{7}$$

Предложение 1. *Если функция $m_k(x)$ задана формулой* (6) *и среди векторов $\{x_i\}_{i=1}^k$ не существует двух совпадающих векторов, то для любой точки $x \in \operatorname{conv}\{x_i\}_{i=1}^k$ выполняются условия*

A1) $m_k(x_i) = f(x_i), \quad i = \overline{1,k}$;

A4) $m_k(x)$ — *липшицева функция*;

A5) $m\big(x, x_1, f(x_1) + c, \ldots, x_k, f(x_k) + c\big) = m\big(x, x_1, f(x_1), \ldots, x_k, f(x_k)\big) + c, \quad k \in \mathbb{N}, \quad c \in \mathbb{R}$.

Доказательство. Условие A1 выполняется в силу условия (7), наложенного на коэффициенты a_{ij}, условие A4 выполняется по построению функции $m_k(x)$, причем $L_m \leqslant \left[\dfrac{\max\limits_{i \neq j} \big|f(x_i) - f(x_j)\big|}{\|x_i - x_j\|}\right]$. Условие A5 также выполняется, поскольку выполняется цепочка равенств

$$m_k(x_p) = \sum_{i=1}^{d+1} a_{ij}\big[f(x_{ij}) + c\big] = \sum_{i=1}^{d+1} a_{ij} f(x_{ij}) + c\sum_{i=1}^{d+1} a_{ij} = f(x_p) + c, \quad p = \overline{1,k}.$$

Что и требовалось доказать.

3.1.2. Кубический RBF-сплайн. Для использования кубического RBF-сплайна [10] требуется, чтобы среди множества векторов $\{x_i\}_{i=1}^k$ существовало $d+1$ линейно независимых векторов. Функция $m_k(x)$ задается следующим образом:

$$m_k(x) = \sum_{i=1}^{k} \lambda_i \varphi\big(\|x - x_i\|\big) + \mu' x + \mu_0. \tag{8}$$

Здесь d — размерность пространства, в котором лежит допустимое множество, $\varphi(r) = r^3$ — ядро сплайна, коэффициенты λ_i, μ_0, μ_1, …, μ_d находятся из решения системы линейных уравнений

$$\begin{pmatrix} 0 & a_{12} & \dots & a_{1k} & 1 & x_{11} & \dots & x_{1d} \\ a_{21} & 0 & \dots & a_{2k} & 1 & x_{21} & \dots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \dots & 0 & 1 & x_{k1} & \dots & x_{kd} \\ 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 \\ x_{11} & x_{22} & \dots & x_{k1} & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{1d} & x_{2d} & \dots & x_{kd} & 0 & 0 & \dots & 0 \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_k \\ \mu_0 \\ \mu_1 \\ \vdots \\ \mu_d \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_k) \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

где $a_{ij} = a_{ji} = \varphi\big(\|x_i - x_j\|\big)$.

Предложение 2. *Если функция $m_k(x)$ задана формулой (8) и среди векторов $\{x_i\}_{i=1}^k$ существуют $d+1$ линейно независимых векторов, то для любой точки $x \in \mathbb{R}^d$ выполняются условия*

A1) $m_k(x_i) = f(x_i), \quad i = \overline{1,k}$;

A4) $m_k(x)$ — *липшицева функция*;

A5) $m\big(x, x_1, f(x_1) + c, \dots, x_k, f(x_k) + c\big) = m\big(x, x_1, f(x_1), \dots, x_k, f(x_k)\big) + c, \quad k \in \mathbb{N}, \quad c \in \mathbb{R}$.

Доказательство. Проверим выполнение условий A1, A4 и A5. Условие A1 выполняется, поскольку сплайн является интерполятором. Поскольку функция дифференцируемая, то она и липшицева, а значит условие A4 выполняется; условие A5 также выполняется, так как правая часть системы разбивается на два слагаемых, тогда и решение также разбивается на две части: решение первой части совпадает с решением для исходной целевой функции, а решение для сдвига можно указать в явном виде

$$\begin{pmatrix} 0 & a_{12} & \dots & a_{1k} & 1 & x_{11} & \dots & x_{1d} \\ a_{21} & 0 & \dots & a_{2k} & 1 & x_{21} & \dots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \dots & 0 & 1 & x_{k1} & \dots & x_{kd} \\ 1 & 1 & \dots & 1 & 0 & 0 & \dots & 0 \\ x_{11} & x_{22} & \dots & x_{k1} & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{1d} & x_{2d} & \dots & x_{kd} & 0 & 0 & \dots & 0 \end{pmatrix} \left[\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_k \\ \mu_0 \\ \mu_1 \\ \vdots \\ \mu_d \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ c \\ 0 \\ \vdots \\ 0 \end{pmatrix} \right] = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_k) \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \begin{pmatrix} c \\ c \\ \vdots \\ c \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Таким образом, условие A5 также верно. Следовательно, предложение 2 доказано.

3.1.3. Логарифмический RBF-сплайн. Для использования логарифмического RBF-сплайна [10] требуется, чтобы среди множества векторов $\{x_i\}_{i=1}^k$ существовало $d+1$ линейно независимых векторов. Функция $m_k(x)$ задается следующим образом: $m_k(x) = \sum\limits_{i=1}^{d+1} \lambda_i \varphi\big(\|x - x_i\|\big) + \mu' x + \mu_0$, где d — размерность пространства, в котором лежит допустимое множество, $\varphi(r) = r^2 \log(r)$ — ядро сплайна, а коэффициенты λ_i, $\mu_0, \mu_1, \dots, \mu_d$ находятся так же, как и для кубического сплайна, только с другим ядром. Для логарифмического RBF-сплайна аналогично кубическому выполняются условия A1, A4 и A5.

3.1.4. Функция на основе обычного кригинга. Для использования функции на основе обычного кригинга требуется, чтобы среди множества $\{x_i\}_{i=1}^k$ не существовало двух совпадающих векторов. Функция $m_k(x)$ задается следующим образом:

$$m_k(x) = \sum_{i=1}^{k} \lambda_i f(x_i). \tag{9}$$

Здесь коэффициенты λ_i определяются как

$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_k \\ \mu \end{pmatrix} = \begin{pmatrix} 1 & c(x_1,x_2) & \dots & c(x_1,x_k) & 1 \\ c(x_2,x_1) & 1 & \dots & c(x_2,x_k) & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c(x_k,x_1) & c(x_k,x_2) & \dots & 1 & 1 \\ 1 & 1 & \dots & 1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} c(x_1,x) \\ c(x_2,x) \\ \vdots \\ c(x_k,x) \\ 1 \end{pmatrix},$$

где $c(x,y) = e^{-\|x-y\|}$.

Предложение 3. *Если функция $m_k(x)$ задана формулой* (9) *и среди векторов $\{x_i\}_{i=1}^k$ не существует двух совпадающих векторов, то для любой точки $x \in \mathbb{R}^d$ выполняются условия*

A1) $m_k(x_i) = f(x_i), \quad i = \overline{1,k}$;

A4) $m_k(x)$ — *липшицева функция*;

A5) $m\big(x, x_1, f(x_1)+c, \dots, x_k, f(x_k)+c\big) = m\big(x, x_1, f(x_1), \dots, x_k, f(x_k)\big) + c, \quad k \in \mathbb{N}, \quad c \in \mathbb{R}$.

Доказательство. Проверим выполнение условий A1, A4 и A5. Условие A1 выполняется, поскольку при $x = x_i$ все коэффициенты λ_i, кроме одного, обращаются в ноль. Условие A4 также выполняется, поскольку функция $m_k(x)$ является дифференцируемой. Условие A5 выполняется, так как для обычного кригинга выполняется соотношение $\sum\limits_{i=1}^{k}\lambda_i = 1$, а значит, при сдвиге целевой функции на константу условие A5 выполняется. Отсюда следует, что предложение 3 доказано.

3.2. Варианты функции для $s_k(x)$.

3.2.1. Кусочно-квадратичная на основе триангуляции Делоне. Для использования кусочно-квадратичной функции на основе триангуляции Делоне [9] требуется, чтобы допустимое множество X содержалось в выпуклой оболочке точек $\{x_i\}_{i=1}^k$. Функция $s_k(x)$ задается следующим образом:

$$s_k(x) = \frac{R_{j(x)}^2 - \|x - x_{j(x)}\|^2}{R_{j(x)}}. \tag{10}$$

Здесь $R_{j(x)}$ — радиус сферы, описанной вокруг симплекса триангуляции Делоне, в который попадает точка x, и $x_{j(x)}$ — центр описанной сферы вокруг симплекса, в который попадает точка x.

Предложение 4. *Если функция $s_k(x)$ задана формулой* (10) *и среди векторов $\{x_i\}_{i=1}^k$ не существует двух совпадающих векторов, то для любой точки $x \in \operatorname{conv}\{x_i\}_{i=1}^k$ выполняются условия*

A2) $s_k(x_i) = 0, \quad i = \overline{1,k}$;

A3) $s_k(x) > 0, \quad x \neq x_i, \quad i = \overline{1,k}$;

A4) $s_k(x)$ — *липшицева функция*;

A6) $s\big(x, x_1, f(x_1)+c, \dots, x_k, f(x_k)+c\big) = s\big(x, x_1, f(x_1), \dots, x_k, f(x_k)\big), \quad k \in \mathbb{N}, \quad c \in \mathbb{R}$,

A7) $s_k(x) \geqslant \min\limits_{i=\overline{1,k}} \|x - x_i\|$.

Доказательство. Условия A2 и A3 выполняются по определению функции $s_k(x)$. Функция является липшицевой внутри каждого симплекса, т.е. условие A4 выполняется. Функция $s_k(x)$ не зависит от значений целевой функции; следовательно, условие A6 выполняется. Проверим выполнение условия A7:

$$\min_{i=\overline{1,k}} \|x - x_i\| = \|x - x_j\| \leqslant \frac{R_{j(x)}^2 - \|x - x_{j(x)}\|^2}{R_{j(x)}}.$$

По теореме косинусов можно записать

$$\|x - x_{j(x)}\|^2 = R_{j(x)}^2 + \|x - x_j\|^2 - 2\|x - x_j\| R_{j(x)}^2 \cos\big(\angle(x - x_j;\, x_{j(x)} - x_j)\big),$$

причем поскольку $x_{j(x)}$ — центр описанной сферы, то $\cos\big(\angle(x - x_j;\, x_{j(x)} - x_j)\big) \geqslant 0$. Тогда можно записать

$$\begin{aligned} &\|x - x_j\| \leqslant \frac{R_{j(x)}^2 - R_{j(x)}^2 - \|x - x_j\|^2 + 2\|x - x_j\| R_{j(x)} \cos\big(\angle(x - x_j;\, x_{j(x)} - x_j)\big)}{R_{j(x)}}, \\ &R_{j(x)} \leqslant -\|x - x_j\| + 2R_{j(x)} \cos\big(\angle(x - x_j;\, x_{j(x)} - x_j)\big), \\ &\|x - x_j\| \leqslant R_{j(x)}\Big(1 + 2\cos\big(\angle(x - x_j;\, x_{j(x)} - x_j)\big)\Big). \end{aligned} \tag{11}$$

Рассмотрим треугольник с вершинами в точках x, x_j, $x_{j(x)}$. У него известны две стороны $\|x - x_j\|$ и $\|x - x_{j(x)}\| = R_{j(x)}$ и угол при вершине x_j, длина третьей стороны не превышает $R_{j(x)}$; следовательно, $\|x - x_j\| \leqslant 2R_{j(x)} \cos\big(\angle(x - x_j;\, x_{j(x)} - x_j)\big)$, что и доказывает неравенство (11) и условие A7. Предложение 4 доказано.

3.2.2. Расстояние до ближайшего измерения. Функция $s_k(x)$ задается следующим образом:

$$s_k(x) = \min_{i=\overline{1,k}} \|x - x_i\|. \tag{12}$$

Предложение 5. *Если функция $s_k(x)$ задана формулой* (12), *то для любой точки $x \in \mathbb{R}^d$ выполняются условия*

A2) $s_k(x_i) = 0, \quad i = \overline{1,k}$;
A3) $s_k(x) > 0, \quad x \neq x_i, \quad i = \overline{1,k}$;
A4) $s_k(x)$ — *липшицева функция*;
A6) $s\big(x, x_1, f(x_1) + c, \ldots, x_k, f(x_k) + c\big) = s\big(x, x_1, f(x_1), \ldots, x_k, f(x_k)\big), \quad k \in \mathbb{N}, \quad c \in \mathbb{R}$,
A7) $s_k(x) \geqslant \min\limits_{i=\overline{1,k}} \|x - x_i\|$.

Доказательство. Проверим выполнение условий A2–A4, A6 и A7. Условия A2 и A3 выполняются по определению функции $s_k(x)$. Функция является липшицевой $L_s = 1$, т.е. условие A4 также выполняется. Функция $s_k(x)$ не зависит от значений целевой функции, следовательно, условие A6 выполняется. Условие A7 выполняется по построению функции $s_k(x)$, что и требовалось доказать.

3.2.3. Гладкая аппроксимация функции расстояния до точки ближайшего испытания. Функция $s_k(x)$ задается следующим образом:

$$s_k(x) = \left[\frac{1}{k}\sum_{i=1}^{k} \|x - x_i\|^{-p}\right]^{-1/p}, \tag{13}$$

где $p \geqslant 1$ — параметр функции.

Предложение 6. *Если функция $s_k(x)$ задана формулой* (13) *и среди векторов $\{x_i\}_{i=1}^k$ не существует двух совпадающих векторов, то для любой точки $x \in \mathbb{R}^d$ выполняются условия*

A2) $s_k(x_i) = 0, \quad i = \overline{1,k}$;
A3) $s_k(x) > 0, \quad x \neq x_i, \quad i = \overline{1,k}$;
A4) $s_k(x)$ — *липшицева функция*;
A6) $s\big(x, x_1, f(x_1) + c, \ldots, x_k, f(x_k) + c\big) = s\big(x, x_1, f(x_1), \ldots, x_k, f(x_k)\big), \quad k \in \mathbb{N}, \quad c \in \mathbb{R}$,
A7) $s_k(x) \geqslant \min\limits_{i=\overline{1,k}} \|x - x_i\|$.

Доказательство. Проверим выполнение условий A2–A4, A6 и A7. Условия A2 и A3 выполняются по определению функции $s_k(x)$, так как (13) можно записать в виде $s_k(x) = \sqrt[p]{k\prod\limits_{i=1}^{k} d_i \left(\sum\limits_{j=1}^{k}\prod\limits_{\substack{i=1\\ j\neq i}}^{k} d_i\right)^{-1}}$, где $d_i = \|x - x_i\|^p$. В связи с тем, что совпадающие векторы в множестве $\{x_i\}_{i=1}^k$ отсутствуют, функция $s_k(x)$ является липшицевой, следовательно, условие A4 выполняется. Функция $s_k(x)$ не зависит от значений целевой функции, а значит, условие A6 также выполняется. Проверим выполнение условия A7:

$$\frac{\min\limits_{i=\overline{1,k}} \|x - x_i\|}{\left[\sum\limits_{i=1}^{k} \|x - x_i\|^{-p}\right]^{-1/p}} = \left[\min_{i=\overline{1,k}} \|x - x_i\|^p \sum_{i=1}^{k} \|x - x_i\|^{-p}\right]^{1/p} = \left[\sum_{i=1}^{k} \frac{\min\limits_{i=\overline{1,k}} \|x - x_i\|^p}{\|x - x_i\|^p}\right]^{1/p} \leqslant k^{1/p}.$$

Следовательно, $\left[\frac{1}{k}\sum\limits_{i=1}^{k}\|x - x_i\|^{-p}\right]^{-1/p} \geqslant \min\limits_{i=\overline{1,k}} \|x - x_i\|$. Таким образом, условие A7 также выполняется. Предложение 6 доказано.

3.2.4. Функция на основе простого кригинга. Для использования функции на основе простого кригинга требуется, чтобы среди множества $\{x_i\}_{i=1}^k$ не существовало двух совпадающих векторов.

Функция $s_k(x)$ задается следующим образом:

$$s_k(x) = \frac{2\operatorname{diam}(X)}{1 - e^{(\sqrt{2}-2)\operatorname{diam}(X)}}\left[1 - \sum_{i=1}^{k} \lambda_i c(x, x_i)\right]. \tag{14}$$

Здесь $\operatorname{diam}(X)$ — диаметр множества X, а коэффициенты λ_i определяются из решения системы уравнений

$$\begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_k \end{pmatrix} = \begin{pmatrix} 1 & c(x_1,x_2) & \dots & c(x_1,x_k) \\ c(x_2,x_1) & 1 & \dots & c(x_2,x_k) \\ \vdots & \vdots & \ddots & \vdots \\ c(x_k,x_1) & c(x_k,x_2) & \dots & 1 \end{pmatrix}^{-1} \begin{pmatrix} c(x_1,x) \\ c(x_2,x) \\ \vdots \\ c(x_k,x) \end{pmatrix}, \tag{15}$$

где $c(x,y) = \exp\{-\|x-y\|\}$.

Предложение 7. *Если функция $s_k(x)$ задана формулой (14) и среди множества $\{x_i\}_{i=1}^k$ не существует двух совпадающих векторов, то для любой точки $x \in \mathbb{R}^d$ выполняются условия*

A2) $s_k(x_i) = 0, \quad i = \overline{1,k}$;

A3) $s_k(x) > 0, \quad x \neq x_i, \quad i = \overline{1,k}$;

A4) $s_k(x)$ — *липшицева функция*;

A6) $s\big(x, x_1, f(x_1) + c, \dots, x_k, f(x_k) + c\big) = s\big(x, x_1, f(x_1), \dots, x_k, f(x_k)\big), \quad k \in \mathbb{N}, \quad c \in \mathbb{R}$,

A7) $s_k(x) \geqslant \min\limits_{i=\overline{1,k}} \|x - x_i\|$.

Доказательство. Проверим выполнение условий A2–A4, A6 и A7. Условия A2 и A3 выполняются по определению функции $s_k(x)$. По своему смыслу, $s_k(x)$ — это дисперсия ошибки определения значения в точке x. Функция $s_k(x)$ является дифференцируемой, поэтому условие A4 выполняется. Функция $s_k(x)$ не зависит от значений целевой функции, а значит, условие A6 также выполняется. Проверим выполнение условия A7. Рассмотрим выражение $s_k(x) = \dfrac{2\operatorname{diam}(X)}{1 - e^{(\sqrt{2}-2)\operatorname{diam}(X)}}\left[1 - r'C^{-1}r\right]$, где r — вектор-столбец из значений $c(x, x_i)$, а матрица C — матрица из (15). Поскольку матрица C является ковариационной матрицей, то она является неотрицательно определенной; следовательно, матрица C^{-1} также неотрицательно определена. Это означает, что выражение $1 - r'C^{-1}r$ является вогнутой функцией по переменным r_i, $i = \overline{1,k}$, а минимум будет достигаться при $r_i = 0$.

С учетом того, что $r_i = e^{-\|x-x_i\|} \in \left[e^{-\operatorname{diam}(X)};\ e^{-\min\limits_{i=\overline{1,k}} \|x-x_i\|}\right]$, т.е. ограничения имеют вид гиперпараллелепипеда, можно записать, что

$$s_k(x) = \frac{2\operatorname{diam}(X)}{1 - e^{(\sqrt{2}-2)\operatorname{diam}(X)}}\left[1 - r'C^{-1}r\right] \geqslant \frac{2\operatorname{diam}(X)}{1 - e^{(\sqrt{2}-2)\operatorname{diam}(X)}}\left[1 - r'_{\min}C^{-1}r_{\min}\right], \tag{16}$$

где $r_{\min}$ — вектор, составленный из элементов $e^{-\min\limits_{i=\overline{1,k}} \|x-x_i\|}$. Другая возможность снизить значение функции $s_k(x)$ — увеличить количество точек, лежащих на сфере с центром в точке x и радиусом $\min\limits_{i=\overline{1,k}} \|x-x_i\|$. Рассмотрим предельный случай, когда во всех точках сферы расположены точки испытания. Требуется определить дисперсию ошибки в центре сферы путем выбора коэффициентов, минимизирующих искомую дисперсию $E\left\{\left(\int\limits_S \lambda(x_r)Y(x_r)\,dx_r - Y(x)\right)^2\right\} \to \min\limits_{\lambda(x)}$,

$$\int\limits_S\int\limits_S \lambda(x_r)\lambda(y_r)\operatorname{cov}\big(Y(y_r), Y(x_r)\big)\,dy_r\,dx_r - 2\int\limits_S \lambda(x_r)\operatorname{cov}\big(Y(y_r), Y(x)\big)\,dx_r + \operatorname{cov}\big(Y(x), Y(x)\big) \to \min_{\lambda(x)},$$

где x_r, y_r — точки на поверхности сферы и x — центр сферы, интегрирование ведется по всей поверхности сферы. Обратим внимание, что задача является симметричной, следовательно функция $\lambda(x)$ также будет симметричной, а значит, $\lambda(x) = \lambda$. Задача запишется следующим образом:

$$\lambda^2\int\limits_S\int\limits_S \operatorname{cov}\big(Y(y_r), Y(x_r)\big)\,dy_r\,dx_r - 2\lambda\int\limits_S \operatorname{cov}\big(Y(y_r), Y(x)\big)\,dx_r + \operatorname{cov}\big(Y(x), Y(x)\big) \to \min_{\lambda}.$$

Решение этой задачи достигается при $\lambda^* = \int\limits_S \operatorname{cov}\big(Y(y_r), Y(x)\big)\, dx_r \left(\int\limits_S \int\limits_S \operatorname{cov}\big(Y(y_r), Y(x_r)\big)\, dy_r\, dx_r \right)^{-1}$.
Значение в точке минимума, с учетом $\operatorname{cov}\big(Y(y_r), Y(x)\big) = \text{const}$, имеет вид

$$1 - \left(\operatorname{cov}\big(Y(y_r), Y(x)\big) \int\limits_S 1\, dx_r \right)^2 \left(\int\limits_S \int\limits_S \operatorname{cov}\big(Y(y_r), Y(x_r)\big)\, dy_r\, dx_r \right)^{-1}.$$

Рассмотрим интеграл в знаменателе отдельно. Разделим внутренний интеграл на две части: по ближней к точке x_r полусфере и по дальней полусфере, тогда

$$\int\limits_S \int\limits_S \operatorname{cov}\big(Y(y_r), Y(x_r)\big)\, dy_r\, dx_r \geqslant \int\limits_S \left[\int\limits_S \frac{q^2}{2}\, dy_r + \int\limits_S \frac{q^{\sqrt{2}}}{2}\, dy_r \right] dx_r, \tag{17}$$

где $q = e^{-\min\limits_{i=\overline{1,k}} \|x - x_i\|}$. С учетом (17) можно записать, что

$$\begin{aligned}
&\frac{2\operatorname{diam}(X)}{1 - e^{(\sqrt{2}-2)\operatorname{diam}(X)}} \left[1 - r'_{\min} C^{-1} r_{\min}\right] \geqslant \\
&\quad \geqslant \frac{2\operatorname{diam}(X)}{1 - e^{(\sqrt{2}-2)\operatorname{diam}(X)}} \left[1 - \left(\operatorname{cov}\big(Y(y_r), Y(x)\big) \int\limits_S 1 dx_r \right)^2 \left(\int\limits_S \int\limits_S \operatorname{cov}\big(Y(y_r), Y(x_r)\big)\, dy_r\, dx_r \right)^{-1} \right] \geqslant \\
&\quad \geqslant \frac{2\operatorname{diam}(X)}{1 - e^{(\sqrt{2}-2)\operatorname{diam}(X)}} \left[1 - \left(\operatorname{cov}\big(Y(y_r), Y(x)\big) \int\limits_S 1 dx_r \right)^2 \left(\int\limits_S \left[\int\limits_S \frac{q^2}{2}\, dy_r + \int\limits_S \frac{q^{\sqrt{2}}}{2}\, dy_r \right] dx_r \right)^{-1} \right] = \\
&\quad = \frac{2\operatorname{diam}(X)}{1 - e^{(\sqrt{2}-2)\operatorname{diam}(X)}} \left[1 - \left(q \int\limits_S 1 dx_r \right)^2 \left(\int\limits_S \left[\int\limits_S \frac{q^2}{2}\, dy_r + \int\limits_S \frac{q^{\sqrt{2}}}{2} dy_r \right] dx_r \right)^{-1} \right] = \\
&\quad = \frac{2\operatorname{diam}(X)}{1 - e^{(\sqrt{2}-2)\operatorname{diam}(X)}} \left[1 - q^2 \left(\frac{q^2}{2} + \frac{q^{\sqrt{2}}}{2} \right)^{-1} \right] = \\
&\quad = \frac{2\operatorname{diam}(X)}{1 - e^{(\sqrt{2}-2)\operatorname{diam}(X)}} \left[\frac{1 - q^{2-\sqrt{2}}}{q^{2-\sqrt{2}} + 1} \right] \geqslant \frac{2\operatorname{diam}(X)}{1 - e^{(\sqrt{2}-2)\operatorname{diam}(X)}} \left[\frac{1 - q^{2-\sqrt{2}}}{2} \right].
\end{aligned} \tag{18}$$

Покажем, что функция $g(q)$ является убывающей при $q \in (0;1)$: $g(q) = \dfrac{-\log(q)}{1 - q^{2-\sqrt{2}}}$. Рассмотрим производную $g'(q) = -\dfrac{1 - q^{2-\sqrt{2}} + (2-\sqrt{2}) q^{2-\sqrt{2}} \log(q)}{q(1 - q^{2-\sqrt{2}})^2}$. Знаменатель положителен, найдем максимум числителя: $q^{2-\sqrt{2}} - (2-\sqrt{2}) q^{2-\sqrt{2}} \log(q) - 1 \to \max$. Найдем производную и приравняем к нулю

$$(2-\sqrt{2}) q^{1-\sqrt{2}} - (2-\sqrt{2}) q^{1-\sqrt{2}} \log(q) - (2-\sqrt{2}) q^{1-\sqrt{2}} = 0.$$

Экстремум может достигаться только в точке $\log(q) = 0$, т.е. при $q = 1$. При $q = 1$ производная $g'(1) = 0$, во всех остальных точках $g'(q) < 0$. Таким образом, на интервале $q \in (0;1)$ функция $g(q)$ является убывающей. С учетом того, что $q = e^{-\min\limits_{i=\overline{1,k}} \|x - x_i\|}$, верно неравенство

$$\frac{2\operatorname{diam}(X)}{1 - e^{(\sqrt{2}-2)\operatorname{diam}(X)}} \left[\frac{1 - q^{2-\sqrt{2}}}{2} \right] \geqslant \min_{i=\overline{1,k}} \|x - x_i\|. \tag{19}$$

Таким образом, с учетом (16), (18), (19) условие A7 также выполняется, что и требовалось доказать.

4. Результаты вычислительных экспериментов по тестированию однородных алгоритмов.

4.1. Условия тестирования алгоритмов. Для тестирования были выбраны следующие тестовые функции: тестовые функции Dixon–Szego [27], тестовые функции, предложенные В. А. Гришагиным [3, 4], и тестовые функции, генерируемые генератором GKLS [19].

Тестовые функции Гришагина задаются формулой

$$f(x) = -\left\{ \left(\sum_{i=1}^{7}\sum_{i=1}^{7} A_{ij}a_{ij}(x) + B_{ij}b_{ij}(x) \right)^2 + \left(\sum_{i=1}^{7}\sum_{i=1}^{7} C_{ij}a_{ij}(x) + D_{ij}b_{ij}(x) \right)^2 \right\}^{1/2}, \qquad (20)$$

где $a_{ij}(x) = \sin(i\pi x_1)\sin(j\pi x_2)$, $b_{ij}(x) = \cos(i\pi x_1)\cos(j\pi x_2)$, коэффициенты A_{ij}, B_{ij}, C_{ij}, D_{ij} — равномерно распределенные величины на отрезке $[-1; 1]$. Всего определено 100 тестовых функций, задаваемых формулой (20). Функции Гришагина доступны по адресу http://si.deis.unical.it/~yaro/Grishagin_web.zip.

Таблица 1

Параметры тестовых функций, генерируемых генератором GKLS

Название	Размерность	r	ρ	Количество локальных минимумов	Точность решения, Δ
GKLS 2D простой	2	0.9	0.2	10	10^{-4}
GKLS 2D сложный	2	0.9	0.1	10	10^{-4}
GKLS 3D простой	3	0.66	0.2	10	10^{-6}
GKLS 3D сложный	3	0.9	0.2	10	10^{-6}
GKLS 4D простой	4	0.66	0.2	10	10^{-6}
GKLS 4D сложный	4	0.9	0.2	10	10^{-6}
GKLS 5D простой	5	0.66	0.3	10	10^{-7}
GKLS 5D сложный	5	0.66	0.2	10	10^{-7}

Тестовые функции, генерируемые генератором GKLS, являются переопределением квадратичного параболоида, который в произвольных местах переопределен таким образом, что по построению возможно точно указать расстояние от глобального минимума до вершины параболоида r, радиус области притяжения глобального минимума ρ и количество локальных минимумов. Для тестирования рассматривались восемь классов, генерируемых генератором (в каждом по 100 тестовых функций), с параметрами, указанными в табл. 1.

Таблица 2

Тестовые функции Dixon–Szego

Название	Размерность	Область определения	Количество локальных минимумов	Количество глобальных минимумов	Значение глобального минимума
Branin	2	$[-5, 10] \times [0, 15]$	3	3	0.398
Goldstein-Price	2	$[-2, 2]^2$	4	1	3
6 hump Camel	2	$[-3, 3] \times [-2, 2]$	6	2	-1.0316
2D Schubert	2	$[-10, 10]^2$	760	18	-186.731
Hartman 3	3	$[0, 1]^3$	4	1	-3.86
Hartman 6	6	$[0, 1]^6$	4	1	-3.32
Shekel 5	4	$[0, 10]^4$	5	1	-10.1532
Shekel 7	4	$[0, 10]^4$	7	1	-10.4028
Shekel 10	4	$[0, 10]^4$	10	1	-10.5363

Одно из существенных различий тестовых функций Гришагина и генерируемых с помощью генератора GKLS заключается в том, что локальные минимумы в функциях Гришагина более предсказуемы, чем в функциях генератора GKLS, в которых координаты локальных минимумов выбираются почти случайно. Основные параметры тестовых функций из набора Dixon–Szego представлены в табл. 2.

Все алгоритмы тестировались на узлах кластера “СКИФ-УРАЛ”: процессор — Intel Xeon E5472, оперативная память — 8 Гб, ОС — SUSE Linux Enterprise Server 10 (P1), компилятор — ICC 10.1 (20080312) с использованием библиотеки MKL 10.0.3.020. Все алгоритмы тестировались в последовательном режиме. Каждому вычислительному процессу выделялось отдельное ядро.

4.2. Тестирование моделей целевых функций для использования в алгоритме глобальной оптимизации. Проведем тестирование для следующих видов функций $m_k(x)$: кусочно-линейная на основе триангуляции Делоне (условная кодировка — “Д”); кубический RBF-сплайн (условная кодировка — “К”); логарифмический RBF-сплайн (условная кодировка — “Л”); на основе обычного кригинга (условная кодировка — “О”).

Рассматривались следующие функции для использования в качестве функции $s_k(x)$: кусочно-квадратичная на основе триангуляции Делоне (условная кодировка — “Д”); расстояние до ближайшего измерения (условная кодировка — “Р”); гладкая аппроксимация функции расстояния до точки ближайшего испытания (условная кодировка — “Г1”, параметр $p = 1$ в (13)); на основе простого кригинга (условная кодировка — “П”); гладкая аппроксимация функции расстояния до точки ближайшего испытания (условная кодировка — “Г3”, параметр $p = 3$ в (13)); гладкая аппроксимация функции расстояния до точки ближайшего испытания (условная кодировка — “Г5”, параметр $p = 5$ в (13)).

Для сравнения были выбраны тестовые функции из набора Dixon–Szego, функции Гришагина и функции генератора GKLS 2D простые. Здесь и далее параметр T алгоритма ОАКР (см. ниже) выбирался равным 0.01.

В табл. 3 представлено количество тестовых задач, решенных однородным алгоритмом с различными моделями целевых функций. Из таблицы следует, что только два набора моделей позволили решить все тестовые задачи: “К”+“Р” и “Л”+“Р” (они выделены жирным шрифтом). Не решили по одной задаче три набора моделей: “К”+“Д”, “К”+“П” и “О”+“Д”. Не решили по две задачи два набора моделей: “Л”+“Д” и “О”+“П”. В основном проблемы возникли при оптимизации четырехмерных функций Shekel. Хуже всего показали себя наборы моделей, в которых в качестве функции $s_k(x)$ были выбраны функции “Г1” и “Г3”.

На всех тестовых наборах в среднем модели “К”+“Р” потребовали меньше испытаний для решения задачи оптимизации, чем модели “Л”+“Р”. В целом можно отметить, что для решения задач оптимизации на функциях Shekel различным наборам моделей потребовалось от 19 до 965 испытаний целевой функции. Для оптимизации функций Hartmann различным моделям потребовалось от 53 до 918 испытаний. Это свидетельствует о необходимости тщательного выбора моделей целевых функций для однородного алгоритма глобальной оптимизации.

Все модели были упорядочены по среднему количеству обращений к целевой функции по возрастанию. Упорядочивание выполнялось для трех тестовых классов. После каждого упорядочивания модели нумеровались числами (вспомогательными рангами) от 1 до 24 в порядке возрастания среднего количества испытаний. В колонке “Ранг” представлена сумма вспомогательных рангов. Этот показатель позволяет сравнивать модели на различных классах тестовых функций с учетом того, что на одном классе в среднем может требоваться значительно больше испытаний, чем на другом. По этому показателю безусловную победу одержал набор моделей “К”+“Р” с рангом 6. Второе место с рангом 13 разделили наборы моделей “К”+“Д” и “К”+“Г5”.

В среднем время выполнения одной итерации составило 6.87 с (минимальное значение — менее разрешения компьютерного таймера, максимальное — 460.8 с при использовании набора моделей “Д”+“Г3” на функции “Hartmann6”). На всех трех тестовых наборах модели “К”+“Р” оказались быстрее, чем набор моделей “Л”+“Р”.

В дальнейшем будем рассматривать алгоритм с моделями “К”+“Р”, которые можно считать одними из самых эффективных из только что рассмотренных. Будем называть этот алгоритм “однородный алгоритм глобальной оптимизации с кубическим сплайном и расстоянием до точки испытания” (ОАКР).

Далее будут представлены результаты сравнения предлагаемого алгоритма с алгоритмами глобальной оптимизации, которые не требуют предварительного задания константы Липшица. Алгоритм будет сравниваться со следующими алгоритмами на основе множественных оценок констант Липшица: алгоритм DIRECT [15, 23], локальная форма алгоритма DIRECT — DIRECTl [18] и алгоритм на адаптивных диагональных кривых (АДК) [26].

Кроме того, будут представлены результаты сравнения с алгоритмами на основе построения поверхности отклика: алгоритм rbfSolve [12], алгоритм EGO [24] и алгоритм ARBFMIP [20].

В заключение тестирования будет рассмотрено сравнение с алгоритмами, предлагающимися сейчас в коммерческих пакетах IOSO NS GT [13] и LGO [25].

4.3. Сравнение предлагаемого алгоритма глобальной оптимизации с алгоритмами гло-

Таблица 3

Сравнение однородных алгоритмов с различными моделями целевых функций

$m_k(x)$	$s_k(x)$	Количество решенных задач оптимизации			Среднее количество обращений к целевой функции				Среднее время решения задачи оптимизации, с		
		Ф. Гришагина	Dixon–Szego	GKLS пр. 2D	Ф. Гришагина	Dixon–Szego	GKLS пр. 2D	Ранг	Ф. Гришагина	Dixon–Szego	GKLS пр. 2D
Д	Д	100	7	97	101.2	376.14	150.6	46	5.42	1350.91	11.07
Д	Р	99	7	100	114.7	188.71	190.9	42	1.35	166.59	1.79
Д	Г1	85	5	85	146.6	354.20	233.5	61	65.38	40475.11	61.97
Д	П	100	6	100	109.3	435.67	145.8	48	14.82	4270.17	27.89
Д	Г3	94	7	93	178.1	425.14	206.1	64	12.48	31270.53	13.53
Д	Г5	100	6	99	179	265.17	189.9	53	8.03	35399.33	7.12
К	Д	100	8	100	66.7	235.75	116	13	1.86	45.18	5.44
К	**Р**	**100**	**9**	**100**	**69**	**68.78**	**118**	**6**	**0.19**	**0.22**	**0.5**
К	Г1	91	8	95	113	163.50	123.7	26	42.18	183.88	27.67
К	П	100	8	100	82.6	332.13	102.5	20	8.94	1565.73	4.85
К	Г3	98	5	100	141.6	177.40	111.2	27	9.62	12.4	3.19
К	Г5	99	7	100	97.3	137.71	104.9	13	2.46	5.98	1.85
Л	Д	100	7	100	77.3	281.00	127.9	23	2.92	427.09	8.36
Л	**Р**	**100**	**9**	**100**	**76.1**	**111.22**	**138.2**	**14**	**0.27**	**1.16**	**1.1**
Л	Г1	93	8	92	98	351.00	143.4	38	24.39	618.07	31.96
Л	П	100	6	100	85.1	207.83	113.6	17	9.14	155.09	6.65
Л	Г3	96	8	99	122.9	246.13	119.6	32	7.89	50.61	4.75
Л	Г5	100	6	100	104.1	97.83	130.4	22	4.29	2.19	3.69
О	Д	100	8	100	80.6	299.38	148.5	32	8.6	4230.78	50.79
О	Р	100	6	100	88.2	149.17	171.9	27	1.4	22.79	9.92
О	Г1	85	4	82	128.5	445.25	220.5	62	1 590.06	26357.91	467.05
О	П	100	7	100	89.5	354.00	139	36	18.37	4147.3	25.32
О	Г3	98	6	97	156.1	297.83	188	52	172.83	579.02	68.11
О	Г5	99	6	100	134.4	373.67	182.1	54	33.56	811.13	47.17

бальной оптимизации на основе множественных оценок константы Липшица. Для сравнения использовался алгоритм на основе адаптивных диагональных кривых [26] с множественными оценками константы Липшица. В качестве тестовых функций использовались два набора тестовых функций: набор из девяти различных функций — набор Dixon–Szego [27] и набор, состоящий из восьми наборов по сто тестовых задач в каждом, генерируемых генератором GKLS [19]. Для каждой тестовой функции известны координаты глобальных минимумов. Алгоритмы исполнялись до тех пор, пока не выполнялось условие остановки

$$\left|x_i^{'} - x_i^*\right| \leqslant \sqrt[d]{\triangle}\,(b_i - a_i), \quad 1 \leqslant i \leqslant d, \tag{21}$$

где x' — точка, где проводилось очередное испытание, x^* — точка глобального минимума, a и b — векторы границ допустимого множества, $\triangle$ — параметр.

Такое условие остановки может быть использовано только при тестировании алгоритмов на специальных тестовых функциях, поскольку требуется явное указание координат глобального экстремума, однако при тестировании это условие позволяет определить алгоритм, наиболее быстро обнаруживший глобальный минимум.

Результаты тестирования алгоритма ОАКР представлены в табл. 4. При тестировании условие остановки выбиралось в виде (21). Курсивом отмечены те алгоритмы, которые смогли за наименьшее количество обращений к целевой функции решить задачу оптимизации с заданной точностью. Как следует из таблицы, предлагаемый алгоритм оказался лучшим на 6 тестовых функциях из 9 рассматриваемых. Данные для DIRECT, DIRECTl и АДК взяты из [26].

Таблица 4

Количество обращений к целевой функции при решении задач оптимизации на тестовых функциях Dixon–Szego

Тестовая функция	Размерность	Δ	DIRECT	DIRECTl	АДК	ОАКР
Branin	2	10^{-4}	41	31	76	*15*
Shubert	2	10^{-4}	19	*15*	59	81
Goldstein-Price	2	10^{-4}	37	*29*	99	199
6 hump camel	2	10^{-4}	105	127	128	*79*
Shekel 5	4	10^{-6}	57	53	208	*23*
Shekel 7	4	10^{-6}	53	45	1465	*33*
Shekel 10	4	10^{-6}	53	45	1449	*25*
Hartmann3	3	10^{-6}	113	79	137	*53*
Hartmann6	6	10^{-7}	144	*78*	4169	89

Особо требуется отметить достаточно большое количество проведенных испытаний для двумерной функции “Goldstein and Price”. Это объясняется тем, что данная функция имеет очень большую область значения — пять порядков, что приводит к большим погрешностям при моделировании ее рельефа. Одним из вариантов улучшения алгоритма в данном случае может быть использование предварительного масштабирования целевой функции, если она является достаточно “плохой”, аналогично тому, как это делается, например, в [24]. Так, при использовании масштабирования $f'(x) = \log_2\bigl(f(x) - f(x^*) + 1\bigr)$ алгоритму ОАКР потребовалось всего лишь 15 испытаний целевой функции.

Недостатки классического набора тестовых функций Dixon–Szego неоднократно отмечались [26]: это и возможность не попасть в область притяжения глобального минимума, и простота поверхности тестовых функций, их слабая адекватность реальным задачам глобальной оптимизации, в которых, как правило, достаточно сложно попасть в зону притяжения глобального экстремума. Для тестирования в более сложных условиях и используется второй набор тестовых функций, построенных с помощью генератора GKLS.

В табл. 5 представлены результаты тестирования алгоритма ОАКР на тестовых функциях генератора GKLS. Данные для АДК взяты из [26]. Снижение затрат вычислено по формуле $\left(1 - \dfrac{\text{ОАКР}}{\text{АДК}}\right)100\%$. При тестировании максимальное количество испытаний, затраченное алгоритмом ОАКР для решения всех задач из предложенного набора функций GKLS, оказалось на 15–66% меньше, чем требовалось алгоритму на основе АДК. В среднем алгоритм ОАКР требовал для решения всех задач из предложенного набора тестовых функций на 31–72% меньше обращений к целевой функции, чем алгоритм на основе АДК на том же наборе тестовых функций.

Однако, как отмечалось ранее, алгоритм ОАКР затрачивает достаточно большой объем вычислений для решения вспомогательной задачи оптимизации, что требует более четкого определения его области применимости. При сравнении алгоритма ОАКР с алгоритмом на основе АДК для каждого класса тестовых функций GKLS вычислялась область применения алгоритма ОАКР на основе среднего количества обращений к целевой функции для сравниваемых алгоритмов, а также с учетом среднего времени, затрачиваемого алгоритмом ОАКР для решения одной задачи оптимизации в рассматриваемом классе тестовых функций. Граница области применения, согласно модели (5), для алгоритма ОАКР для тестового класса “GKLS 2D простой” — 8.32 мс, для класса “GKLS 2D сложный” — 8.20 мс; для остальных классов гра-

Таблица 5

Сравнение предлагаемого алгоритма с алгоритмами на основе множества констант Липшица на тестовых функциях GKLS

	Максимальное число испытаний			Среднее число испытаний		
Класс тестовых функций	АДК	ОАКР	Снижение затрат	АДК	ОАКР	Снижение затрат
GKLS 2D простой	403	341	15%	176.25	121.07	31%
GKLS 2D сложный	1809	838	46%	675.74	360.81	47%
GKLS 3D простой	2506	1549	62%	735.76	350.11	52%
GKLS 3D сложный	6006	3943	66%	2006.82	915.52	54%
GKLS 4D простой	14520	6245	43%	5014.13	1810.8	64%
GKLS 4D сложный	42649	13828	32%	16473.02	5050.9	69%
GKLS 5D простой	33533	12072	36%	5129.85	2320.98	55%
GKLS 5D сложный	93745	29515	31%	30471.83	8407.89	72%

ница области применения для сложного класса, как правило, превышала границу для простого класса примерно в два раза: так, для трехмерных задач эти границы были 14.73 мс и 30.59 мс для простого и сложного классов соответственно, для четырехмерного класса — 148.40 мс и 277.94 мс соответственно, а для пятимерного — 102.57 мс и 246.44 мс соответственно. Снижение границы области применения для пятимерного класса тестовых функций связано с выбором параметра Δ в (21): так, для четырехмерного случая правая часть равна 0.0316, а для пятимерного тестового класса значение правой части задавалось равным 0.0398. В целом по результатам тестирования для тестовых функций GKLS можно заключить, что применение алгоритма ОАКР является предпочтительным, если время вычисления целевой функции более 278 мс согласно модели (5).

По результатам оптимизации для минимизаций функций из набора Dizon–Szego алгоритму ОАКР не потребовалось задействовать алгоритм глобальной оптимизации, в среднем на каждой итерации алгоритм ОАКР использовал 1.43 запуска процедур локальной оптимизации. Коэффициент достоверности не изменялся и остался равным 1.0.

При решении задач из тестового набора GKLS в среднем на одну итерацию требовалось от 2.91 до 10.91 процедур локальной оптимизации. Процедура глобальной оптимизации требовалась значительно реже — от 0.0011 до 0.0950 запусков (т.е. она использовалась от одного запуска на почти 10 итераций до одного запуска на почти 880 итераций). Коэффициент достоверности в среднем был в пределах 1.0–1.1 (за исключением класса “GKLS 2D сложный” — 3.2). С увеличением размерности тестовых задач максимальный коэффициент достоверности снижался.

4.4. Сравнение предлагаемого алгоритма глобальной оптимизации с алгоритмами глобальной оптимизации на основе методологии поверхностей отклика. В настоящее время наиболее экономичными считаются алгоритмы глобальной оптимизации на основе поверхности отклика. Для тестирования выбраны алгоритмы, предложенные в [12, 20, 24]. Условие остановки задавалось в виде

$$\frac{f(x_k) - f(x^*)}{|f(x^*)|} \leqslant \varepsilon, \tag{22}$$

где x_k — текущая точка испытания, x^* — точка глобального минимума, ε — заданная погрешность. Погрешность была выбрана $\varepsilon = 0.01$. Для тестирования использовались алгоритмы глобальной оптимизации, реализованные в пакете TOMLAB© [21]. Условием окончания работы алгоритмов было выбрано либо достижение 1000 итераций, либо выполнение условия остановки (22). Первоначально точки испытаний располагались во всех вершинах допустимого множества. Остальные параметры выбирались по умолчанию. Сравнивались пять алгоритмов:

1) алгоритм rbfSolve [12], вспомогательный алгоритм glcFast (DIRECT [23]; условная кодировка — “RBF1”);

2) алгоритм rbfSolve [12], вспомогательный алгоритм glcCluster (DIRECT [23] + алгоритм кластеризации; условная кодировка — “RBF2”);

3) алгоритм EGO [24], вспомогательный алгоритм glcFast (DIRECT [23]; условная кодировка — “EGO1”);

4) алгоритм EGO [24], вспомогательный алгоритм glcCluster (DIRECT [23] + алгоритм кластеризации;

условная кодировка — “EGO2”);

5) алгоритм ARBFMIP [20] (условная кодировка — “ARBF”).

Результаты сравнения алгоритма OAKP с алгоритмами на основе поверхностей отклика представлены в табл. 6 и 7. В качестве условия остановки для алгоритма OAKP выбиралось условие (22). В табл. 6 жирным шрифтом выделены те алгоритмы, которые смогли найти глобальный минимум. Курсивом отмечены алгоритмы, которые потребовали минимального количества испытаний для решения задачи с необходимой точностью. В четырех случаях из девяти алгоритм OAKP потребовал меньше всего испытаний для минимизации тестовых функций. Существенно хуже алгоритм OAKP показал себя на тестовой функции “Goldstein and Price”. Причины такого поведения обсуждались в предыдущем тестировании.

Таблица 6

Количество обращений к целевой функции алгоритмов из пакета TOMLAB на наборе тестовых функций Dixon–Szego

Тестовая функция	Количество обращений к целевой функции					
	OAKP	RBF1	RBF2	EGO1	EGO2	ARBF
Branin RCOS	**26**	**49**	**38**	**21**	*14*	**39**
2D Shubert	*45*	**170**	**192**	6	382	**233**
Goldstein and Price	**1362**	**203**	**183**	*50*	**58**	**188**
Six-Hump Camel	**59**	**22**	**22**	**47**	**81**	*13*
Shekel 5	*32*	1000	1000	91	**60**	**60**
Shekel 7	*41*	**53**	**54**	94	108	**74**
Shekel 10	*43*	**54**	**58**	77	61	**71**
Hartman 3	**40**	**28**	**28**	*23*	**74**	**41**
Hartman 6	**95**	**71**	*66*	**83**	**77**	**88**

Среднее время минимизации функций из тестового набора Dixon–Szego существенно отличалось для различных алгоритмов: так, алгоритмы RBF1 и RBF2 потребовали 1209 с и 1295 с соответственно, алгоритмы EGO1 и EGO2 потребовали 147 и 1740 с соответственно (алгоритм EGO2 решал двумерную задачу 2D Shubert около 3 часов 50 минут), алгоритм ARBF затратил в среднем 479 с. Предлагаемый алгоритм OAKP потратил в среднем 2 с, причем основное время было затрачено на решение задачи “Goldstein and Price” — 13 с.

Таблица 7

Результаты тестирования алгоритмов из пакета TOMLAB на тестовых функциях Гришагина

Критерии сравнения	OAKP	RBF1	RBF2	EGO1	EGO2	ARBF
Среднее количество испытаний	76.92	286.82	286.67	36.81	38.09	87.99
Максимальное количество испытаний	349	1000	1000	127	97	225
Среднее время оптимизации, с	0.17	2563.40	1629.96	81.54	36.20	521.37
Количество найденных глобальных минимумов	100	89	90	40	38	98
Граница области применения алгоритма OAKP, мс	—	0.8	0.8	—	—	15.1

В табл. 7 представлены результаты оптимизации на ста тестовых функциях Гришагина. Алгоритм OAKP — единственный алгоритм, который решил все сто задач оптимизации. Алгоритм ARBF смог решить 98 задач, алгоритм RBF — 90, а алгоритм EGO — 40.

По среднему времени оптимизации безусловным лидером является алгоритм OAKP, хотя надо отметить, что алгоритмы TOMLAB частично реализованы в системе MATLAB, что может снижать их производительность. По среднему числу испытаний лидирует алгоритм EGO, однако с учетом всего лишь 40 найденных экстремумов этот алгоритм целесообразно исключить из рассмотрения. В этом случае лучшим окажется алгоритм OAKP, затем с небольшим отрывом следует алгоритм ARBF. По максимальному

числу испытаний также лидирует алгоритм EGO, но исключив его из рассмотрения, лидером станет алгоритм ARBF, на втором месте будет алгоритм OAKP.

В целом можно заключить, что характеристики алгоритма OAKP являются сопоставимыми с алгоритмами, основанными на построении поверхности отклика. Область применения алгоритма OAKP по сравнению с алгоритмами RBF1, RBF2, ARBF рассчитывалась на основе среднего количества затрачиваемых испытаний алгоритмами OAKP, RBF1, RBF2, ARBF и среднего времени работы алгоритма OAKP (табл. 7). Для алгоритмов EGO1 и EGO2 граница области применения не рассчитывалась, так как алгоритм показал низкий процент решения тестовых задач. В целом при сравнении алгоритма OAKP с алгоритмами на основе поверхностей отклика можно заключить, что область применения алгоритма OAKP — целевые функции с временем выполнения более 16 мс.

4.5. Сравнение предлагаемого алгоритма глобальной оптимизации с алгоритмами, реализованными в программных комплексах IOSO и LGO. Сравним алгоритм OAKP с широко распространенными алгоритмами, реализованными в коммерческих пакетах IOSO NS GT и LGO.

Алгоритмы тестировались на тестовых функциях (20), предложенных Гришагиным, и тестовых функциях, генерируемых генератором GKLS [19] с параметрами, указанными в табл. 1. Условие остановки было выбрано (21). Параметры алгоритма IOSO выбирались по умолчанию. Менялась только размерность пространства в соответствии с выбранной тестовой задачей.

Для пакета LGO тестировалось четыре алгоритма, реализованные в пакете: LGO1 — локальный поиск, LGO2 — глобальный поиск на основе метода ветвей и границ + локальный поиск, LGO3 — случайный глобальный поиск + локальный поиск, LGO4 — случайный мультистарт + локальный поиск. Параметры алгоритма LGO задавались следующим образом: “Penalty Multiplier”=1, “Random Seed”=1, “Time Limit (Integer Secs)”=3600, “Local Search Tolerance”=10^{-3}. Параметр “Global Search Function Calls” для функций Гришагина задавался равным 2000, для двумерных функций GKLS — 10 000, для трехмерных — 30 000, для четырехмерных — 50 000 и для пятимерных — 100 000.

Таблица 8

Количество решенных задач оптимизации алгоритмами IOSO и LGO

Тестовый класс	OAKP	IOSO	LGO1	LGO2	LGO3	LGO4
Функции Гришагина	100	**19**	**8**	64	79	94
GKLS 2D простой	100	**16**	**10**	95	99	100
GKLS 2D сложный	100	**4**	**2**	52	89	99
GKLS 3D простой	100	**5**	**3**	72	82	98
GKLS 3D сложный	100	**3**	**0**	54	74	91
GKLS 4D простой	100	**3**	**1**	**24**	55	88
GKLS 4D сложный	100	**0**	**1**	**15**	**43**	76
GKLS 5D простой	100	**7**	**0**	**37**	59	90
GKLS 5D сложный	100	**0**	**0**	**17**	**43**	64
В среднем решено:	*100*	*8.14*	*4.17*	*47.78*	*69.22*	*88.89*

В табл. 8 представлены результаты решения тестовых задач. Алгоритм OAKP решил все 100 задач в каждом тестовом классе. Алгоритм IOSO в среднем решал около 8 задач из 100 в каждом тестовом классе, что оказалось лучше, чем у алгоритма LGO1 — примерно 4 задачи из 100, и хуже, чем алгоритмы LGO3 и LGO4 — 69 и 89 из 100 тестовых задач соответственно. Заметим, что у алгоритмов IOSO и LGO1 есть классы тестовых функций, для которых алгоритмы не смогли решить ни одной задачи.

В табл. 9 представлено среднее количество испытаний, затраченное на оптимизацию тестовых функций из соответствующего класса. Символом “—” отмечены классы функций, в которых алгоритмы не смогли решить ни одной задачи, а жирным шрифтом — классы функций, в которых было решено менее 50 тестовых задач. Если рассматривать эту таблицу без учета алгоритмов, которые не смогли решить более 50 тестовых задач оптимизации, то меньше всего испытаний по каждому тестовому классу потребовалось алгоритму OAKP.

Область применения алгоритма OAKP относительно алгоритмов LGO1 и IOSO не определялась, так как на представленных тестовых функциях эти алгоритмы не смогли решить более 50 из 100 тестовых задач. Граница области применения для алгоритма OAKP по сравнению с алгоритмами IOSO и LGO

Таблица 9

Среднее количество испытаний для минимизации тестовых функций алгоритмами IOSO и LGO

Тестовый класс	Алгоритмы					
	OAKP	IOSO	LGO1	LGO2	LGO3	LGO4
Функции Гришагина	69.20	**23.37**	**25.13**	413.22	909.68	517.00
GKLS 2D простой	121.07	**32.19**	**16.20**	1 162.47	1 838.02	761.69
GKLS 2D сложный	360.81	**30.50**	**22.00**	1 094.15	1 655.64	1 585.36
GKLS 3D простой	350.11	**290.60**	**33.33**	5 145.94	11 061.26	3 700.23
GKLS 3D сложный	915.52	**319.00**	—	5 456.20	11 708.01	5 604.37
GKLS 4D простой	1 810.80	**326.67**	**23.00**	**4 500.25**	12 111.80	10 950.60
GKLS 4D сложный	5 050.90	—	**23.00**	**5 346.33**	**9 116.98**	14 777.54
GKLS 5D простой	2 320,98	**446.43**	—	**17740.03**	34 252.15	22 928.01
GKLS 5D сложный	8 407,89	—	—	**10 018.06**	**30 992.30**	41 045.97

рассчитывалась на основе среднего количества испытаний рассматриваемых алгоритмов (OAKP, LGO2, LGO3, LGO4) и среднего времени работы алгоритма OAKP. По сравнению с алгоритмом LGO2 область применения алгоритма OAKP — целевые функции, время вычисления которых более 7.35 мс, при этом на тестовых классах, в которых удалось решить менее 50% предложенных задач, область применения алгоритма OAKP — целевые функции с временем вычисления до 10.7 с. По сравнению с алгоритмом LGO3 область применения алгоритма OAKP — целевые функции с временем вычисления 46.1 мс, по сравнению с LGO4 — 326.4 мс.

5. Заключение. Предложены схемы ускорения однородных алгоритмов глобальной оптимизации. Предложены и протестированы модели целевых функций, которые могут быть использованы в однородных алгоритмах глобальной оптимизации. Было показано, что для различных задач требуется использование различных моделей. В среднем из предложенных моделей целесообразно рассматривать однородные алгоритмы с кубическим или логарифмическим сплайном и минимальным расстоянием до точки, где было проведено испытание. Тестирование алгоритма OAKP по сравнению с различными классами алгоритмов глобальной оптимизации продемонстрировало эффективность предложенного подхода к построению алгоритма глобальной оптимизации. По сравнению с алгоритмами на основе множественных оценок константы Липшица предложенный алгоритм в среднем эффективнее примерно на 56% по количеству испытаний на тестовых функциях GKLS. По сравнению с алгоритмами на основе поверхности отклика предложенный алгоритм в среднем требует на 14% меньше испытаний на тестовых функциях Гришагина. В сравнении с алгоритмами IOSO и LGO предложенный алгоритм требует в среднем в 5–10 раз меньше испытаний. Область применения предлагаемого алгоритма — целевые функции с временем вычисления более 327 мс.

СПИСОК ЛИТЕРАТУРЫ

1. *Ананченко А.Г.* Разработка алгоритмов и программных комплексов для глобальной оптимизации химико-технологических систем. Дисс. ... канд. техн. наук. Санкт-Петербург, 2004.
2. *Антонов М.О., Елсаков С.М., Ширяев В.И.* Нахождение оптимального расположения радиомаяков в разностно-дальномерной системе посадки летательного аппарата // Авиакосмическое приборостроение. 2005. № 11. 41–45.
3. *Гришагин В.А.* Исследование одного класса численных методов решения задач многоэкстремальной оптимизации. Дисс. ... канд. физ.-мат. наук. Горький, 1983.
4. *Гришагин В.А.* Операционные характеристики некоторых алгоритмов глобального поиска // Проблемы случайного поиска. Рига: Зинатне, 1978. № 7. 198–206.
5. *Евтушенко Ю.Г.* Численный метод поиска глобального экстремума функций (перебор на неравномерной сетке) // Журн. вычисл. матем. и матем. физ. 1971. **11**, № 6. 1390–1400.
6. *Елсаков С.М., Ширяев В.И.* Однородные алгоритмы многоэкстремальной оптимизации // Журн. вычисл. матем. и матем. физ. 2010. **50**, № 10. 1–14.
7. *Квасов Д.Е., Сергеев Я.Д.* Многомерный алгоритм глобальной оптимизации на основе адаптивных диагональных кривых // Журн. вычисл. матем. и матем. физ. 2003. **43**, № 1. 42–59.
8. *Пиявский С.А.* Один алгоритм отыскания абсолютного экстремума функции // Журн. вычисл. матем. и матем. физ. 1972. **12**, № 12. 57–67.

9. *Препарата Ф., Шеймос М.* Вычислительная геометрия: Введение. М.: Мир, 1989.
10. *Роженко А.И.* Теория и алгоритмы вариационной сплайн-аппроксимации. Новосибирск: ИВМиМГ, 2005.
11. *Стронгин Р.Г.* Численные методы в многоэкстремальных задачах. М.: Наука, 1978.
12. *Björkman M., Holmström K.* Global optimization of costly nonconvex functions using radial basis functions // Optimization and Engineering. 2000. N 4. 373–397.
13. *Egorov I.N., Kretinin G.V., Leshchenko I.A.* Robust design optimization strategy of IOSO technology // Proc. Fifth World Congress on Computational Mechanics. Vienna, Austria. 2002. 1–8.
14. *Elsakov S.M., Shiryaev V.I.* Linear homogeneous algorithms of global optimization // Global Optimization: Theory, Methods & Applications. Series Lecture Notes in Decision Science. Vol. 12. Hong Kong, London, Tokyo: Global-Link Publisher, 2009. 241–247.
15. *Finkel D.E.* Global optimization with the direct algorithm. PhD thesis. Raleigh, NC, 2005.
16. *Floudas C.A.* Deterministic global optimization: theory, methods and its applications. New York: Springer, 1999.
17. *Floudas C.A., Gounaris C.E.* A review of recent advances in global optimization // J. of Global Optimization. 2009. N 1. 3–38.
18. *Gablonsky J.M., Kelley C.T.* A locally-biased form of the direct algorithm // J. of Global Optimization. 2001. N 1. 27–37.
19. *Gaviano M., Kvasov D.E., Lera D., Sergeyev Ya.D.* Generation of classes of test functions with known local minima // ACM Trans. Math. Soft. 2003. N 4. 469–480.
20. *Holmström K.* An adaptive radial basis algorithm (ARBF) for expensive black-box global optimization // J. of Global Optimization. 2008. N 3. 447–464.
21. *Holmström K., Göran A.O., Edvall M.M.* User's guide for TOMLAB 7 (http://tomopt.com/docs/TOMLAB.pdf).
22. *Jones D.R.* A taxonomy of global optimization methods based on response surfaces // J. of Global Optimization. 2001. N 4. 345–383.
23. *Jones D.R., Perttunen C.D., Stuckman B.E.* Lipschitzian optimization without the Lipschitz constant // J. Optim. Theory Appl. 1993. N 1. 157–181.
24. *Jones D.R., Schonlau M., Welch W.J.* Efficient global optimization of expensive black-box functions // J. of Global Optimization. 1998. N 4. 455–492.
25. *Pintér J.D.* Nonlinear optimization with GAMS/LGO // J. of Global Optimization. 2007. N 1. 79–101.
26. *Sergeyev Y.D., Kvasov D.E.* Global search based on efficient diagonal partitions and a set of Lipschitz constants // SIAM J. on Optimization. 2006. N 3. 910–937.
27. Towards global optimization 2 / Eds. G.P. Szego, L.C.W. Dixon. New York : North-Holland Pub., 1978.
28. *Xiaojun W., Yu M., Xia W.Q.* Implicit fitting and smoothing using radial basis functions with partition of unity // Proc. Ninth Int. Conf. on Computer Aided Design and Computer Graphics. Washington: IEEE Computer Society, 2005. 139–148.

Поступила в редакцию
03.11.2010