БЫСТРЫЙ ПОИСК НУЛЯ ВЫПУКЛОЙ КУСОЧНО-ЛИНЕЙНОЙ ФУНКЦИИ*

Е. В. Просолупов
e.prosolupov@spbu.ru

Г. Ш. Тамасян
g.tamasyan@spbu.ru

10 сентября 2015 г.

Аннотация. Известно, что задача ортогонального проектирования точки на стандартный симплекс ставится, как задача квадратичного программирования. В работе Held, Wolfe, Crowder (1974) было впервые представлено сведéние данной задачи к решению скалярного уравнения. Позднее P. Brucker (1984) предложил алгоритм для решения более общей задачи (квадратичного программирования), имеющий линейный порядок сложности.

В докладе анализируется конструктивный алгоритм по поиску нуля скалярного уравнения предложенный, N. Maculan, G. G. de Paula Jr (1989). На идейном уровне он близок к работе Brucker. Детально исследован вопрос об его трудоемкости, а также приведены результаты численных экспериментов.

1°. Пусть задано множество вещественных чисел $\{c_1, \ldots, c_n\}$ и положительная константа β. Обозначим $N = 1 : n$. Рассмотрим функцию φ от скалярной переменной t:

$$\varphi(t) = \sum_{j \in N} \max\{c_j - t, 0\}. \tag{1}$$

Задача. Решить уравнение

$$\varphi(t) = \beta. \tag{2}$$

Нетрудно заметить, что $\varphi(t)$ — непрерывная кусочно-линейная выпуклая функция на всей вещественной оси (см. рис. 1).

Более того при $t \leqslant \max\limits_{j \in N} c_j$ функция $\varphi(t)$ монотонно убывает и равна нулю при $t \geqslant \max\limits_{j \in N} c_j$. Таким образом, уравнение (2) имеет решение и это решение единственно. Обозначим его t^*.

*Семинар по конструктивному негладкому анализу и недифференцируемой оптимизации «CNSA & NDO»: `http://www.apmath.spbu.ru/cnsa/`

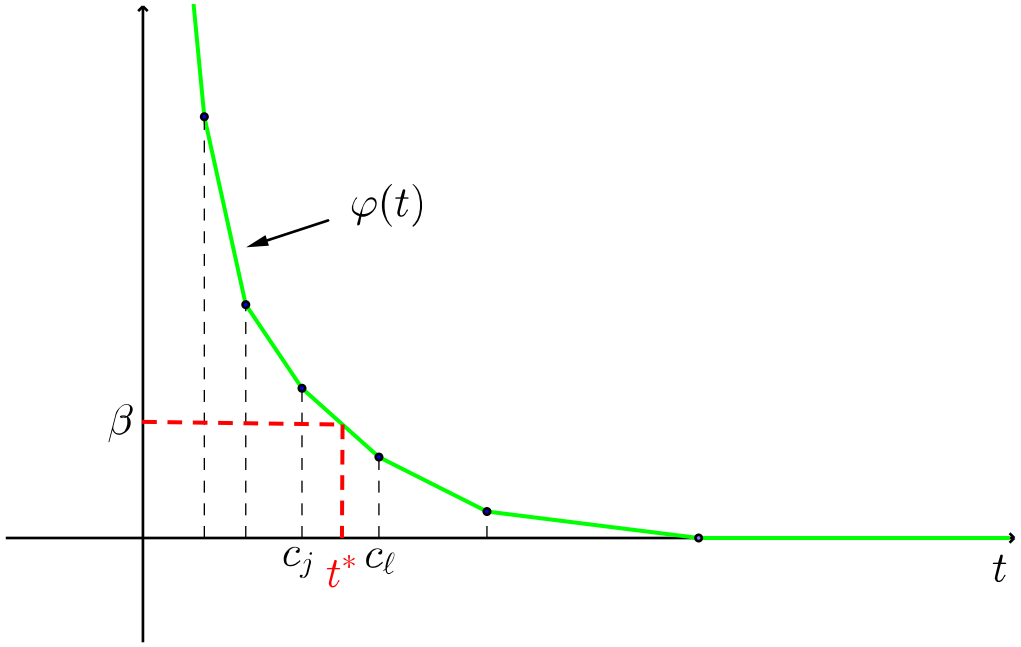

Рис. 1. График функции $\varphi(t)$

При $\beta = 1$ задача (2) возникает в процессе поиска ортогональной проекции точки на стандартный симплекс (см. [1–8]).

Подробно изучим алгоритм решения уравнения (2), описанный в работе [5].

2°. Алгоритм Maculan – de Paula (Maculan). Далее нам потребуется понятие медианы множества.

ОПРЕДЕЛЕНИЕ 1. Пусть $S = \{s_1, \ldots, s_\ell\}$ — множество вещественных чисел. Обозначим упорядоченный по неубыванию набор элементов множества S через $(a_1, \ldots, a_\ell)$. Под медианой M множества S понимается величина $a_m \in S$, которая стоит на позиции $m = \lfloor \frac{\ell+1}{2} \rfloor$, т. е.

$$M = a_m,$$

где $\lfloor z \rfloor$ — целая часть числа z.

Переходим к описанию алгоритма.

Инициализация.

$$S := \{c_1, \ldots, c_n\}, \quad J := N,$$
$$v := 0, \quad p := 0, \quad q := 0.$$

Введём функцию

$$\Phi(t, x, L, v, p, q) := \sum_{j \in L} (x_j - t) + v + p(q - t), \tag{3}$$

где $x = (x_1, \ldots, x_n)$, $L \subseteq N$.

Шаг 1. Имеем множество $S = \{c_j \mid j \in J\}$. Найдём медиану M множества S и построим следующие индексные множества

$$L_= := \{j \in J \mid c_j = M\},$$
$$L_> := \{j \in J \mid c_j > M\},$$
$$L_< := \{j \in J \mid c_j < M\}.$$

Пусть $m \in L_=$, т. е.

$$c_m = M. \tag{4}$$

Вычислим

$$z = \Phi(M, c, L_>, v, p, q). \tag{5}$$

Если $z \geqslant \beta$, то переходим к **Шагу 2**; иначе к **Шагу 3**.

Шаг 2. Положим

$$J := L_> \cup \{m\}, \qquad S := \{c_j \mid j \in J\},$$

где m — индекс найденный на **Шаге 1** (см. (4)).

Если $|J| \geqslant 3$, то переходим к **Шагу 1**; иначе при $v = z$ и $q = M$ — к **Шагу 4**. Здесь $|J|$ — мощность индексного множества J.

Шаг 3. Положим

$$J := L_< \cup \{m\}, \qquad v := z, \qquad q := M,$$
$$p := p + |L_=| - 1 + |L_>|.$$

Если $|J| \geqslant 2$, то переходим к **Шагу 1**; иначе к **Шагу 4**.

Шаг 4. Решение уравнения (2) находим по формуле

$$t^* := q - \frac{\beta - v}{1 + p}. \tag{6}$$

Описание алгоритма завершено.

Замечание 1. Если индексное множество $L_=$ состоит более чем из одного элемента, то в качестве c_m в формуле (4) можно выбрать любой элемент множества S равный медиане M.

Замечание 2. Величина z вычисляемая на **Шаге 1** (см. (5)) является значением функции $\varphi(t)$ (см. (1)) в точке t равной медиане M, т. е.

$$\Phi(M, c, L_>, v, p, q) = \varphi(M).$$

Замечание 3. Мощность индексного множества J от итерации к итерации строго убывает (практически в два раза). Это гарантирует конечность процесса.

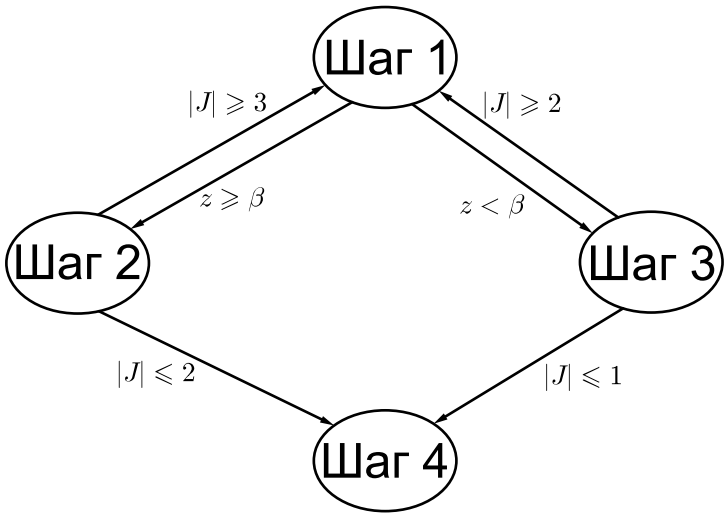

Рис. 2. Схема алгоритма

На рис. 2 приведена схема работы алгоритма Maculan. Каждая итерация алгоритма кроме последней состоит из последовательного выполнения Шага 1+Шага 2, или Шага 1+Шага 3 с последующим возвратом к Шагу 1. Переход от Шага 1 к Шагу 2 осуществляется, если значение функции φ в текущей медиане c_m больше или равно β и требуется продолжить поиск в правой половине списка c_j ($c_j \geqslant c_m$). Переход от Шага 1 к Шагу 3 осуществляется, в противном случае. Тогда поиск продолжается в левой части набора ($c_j \leqslant c_m$). Если будет выполнен переход к Шагу 4, то, после его выполнения, алгоритм завершит свою работу.

Отметим, что приведенный алгоритм имеет линейный порядок сложности. Известно, что сложность поиска медианы в числовом множестве из n элементов составляет $O(n)$ [9]. Вычисление функции $\Phi(t, x, L, v, p, q)$ имеет трудоемкость $O(|L|)$. Таким образом все действия Шага 1 имеют порядок сложности $O(|J|)$. Если обозначить n_i мощность множества J на i-й итерации, то согласно описанию Шага 2 и Шага 3 получаем

$$n_{i+1} \leqslant \frac{n_i + 1}{2} = \frac{n_i}{2} + \frac{1}{2}.$$

То есть, количество действий на всех итерациях алгоритма имеет порядок

$$O(n) + O\left(\frac{n}{2} + \frac{1}{2}\right) + O\left(\frac{n}{4} + \frac{3}{4}\right) + O\left(\frac{n}{8} + \frac{7}{8}\right) + \ldots + O(2) = O(n).$$

3°. Примеры.

ПРИМЕР 1. Найти решение уравнения (2) при $\beta = 1$ и $c_j = 2j - 1$, $j = 1 : 6$.

Инициализация. Имеем $J = 1 : 6$, $S = \{1, 3, 5, 7, 9, 11\}$, $v = 0$, $p = 0$, $q = 0$.

Шаг 1. Находим медиану множества S: $M = 5$. Поэтому $m = 3$ (см. (4)). Строим индексные множества:

$$L_= = \{3\}, \qquad L_> = \{4, 5, 6\}, \qquad L_< = \{1, 2\}.$$

Вычисляем z (см. (5)):

$$z = (7 - 5) + (9 - 5) + (11 - 5) + 0 + 0(0 - 5) = 12.$$

Так как $z \geqslant 1$, то переходим к **Шагу 2**.

Шаг 2. Имеем $J = L_> \cup \{m\} = \{3, 4, 5, 6\}$. Тогда $S = \{5, 7, 9, 11\}$. Так как мощность множества J равна 4, то переходим к **Шагу 1**.

Шаг 1.

$$M = 7, \qquad m = 4,$$
$$L_= = \{4\}, \qquad L_> = \{5, 6\}, \qquad L_< = \{3\},$$
$$z = (9 - 7) + (11 - 7) + 0 + 0(0 - 7) = 6.$$

Шаг 2.

$$J = L_> \cup \{m\} = \{4, 5, 6\},$$
$$S = \{7, 9, 11\}.$$

Шаг 1.

$$M = 9, \qquad m = 5,$$
$$L_= = \{5\}, \qquad L_> = \{6\}, \qquad L_< = \{4\},$$
$$z = (11 - 9) + 0 + 0(0 - 7) = 2.$$

Шаг 2.

$$J = L_> \cup \{m\} = \{5, 6\},$$
$$S = \{9, 11\}.$$

Так как $|J| = 2$, то переходим к последнему шагу.

Шаг 4. Имеем $p = 0$, $q := M = 9$, $v := z = 2$. Тогда искомое решение вычисляется по формуле (см. (6)):

$$t^* = 9 - \frac{1 - 2}{1 + 0} = 10.$$

ПРИМЕР 2. Найти решение уравнения (2) при $\beta = 1$ и

$$c_j = \begin{cases} 1, & j = 1:5; \\ 2, & j = 6:7; \\ 3, & j = 8:9. \end{cases}$$

Инициализация. Имеем $J = 1:9$, $S = \{1,1,1,1,1,2,2,3,3\}$, $v = 0$, $p = 0$, $q = 0$.

Шаг 1. Находим медиану множества S: $M = 1$. Поэтому $m = 5$ (см. (4)). Строим индексные множества:

$$L_= = \{1,2,3,4,5\}, \qquad L_> = \{6,7,8,9\}, \qquad L_< = \emptyset.$$

Вычисляем z (см. (5)):

$$z = (2-1) + (2-1) + (3-1) + (3-1) + 0 + 0(0-1) = 6.$$

Так как $z \geqslant 1$, то переходим к **Шагу 2**.

Шаг 2. Имеем $J = L_> \cup \{m\} = \{5,6,7,8,9\}$. Тогда $S = \{1,2,2,3,3\}$. Так как мощность множества J равна 5, то переходим к **Шагу 1**.

Шаг 1.

$$M = 2, \qquad m = 7,$$
$$L_= = \{6,7\}, \qquad L_> = \{8,9\}, \qquad L_< = \{5\},$$
$$z = (3-2) + (3-2) + 0 + 0(0-2) = 2.$$

Шаг 2.

$$J = L_> \cup \{m\} = \{7,8,9\},$$
$$S = \{2,3,3\}.$$

Шаг 1.

$$M = 3, \qquad m = 8,$$
$$L_= = \{8,9\}, \qquad L_> = \emptyset, \qquad L_< = \{7\},$$
$$z = 0 + 0 + 0(0-3) = 0.$$

Так как $z < 1$, то переходим к **Шагу 3**.

Шаг 3.

$$J = L_< \cup \{m\} = \{7,8\}, \qquad S = \{2,3\},$$
$$v := z = 0, \qquad q := M = 3, \qquad p := 0 + 2 - 1 + 0 = 1.$$

Так как $|J| = 2$, то переходим к **Шагу 1**.

Шаг 1.

$$M = 2, \qquad m = 7,$$
$$L_= = \{7\}, \qquad L_> = \{8\}, \qquad L_< = \emptyset,$$
$$z = (3-2) + 0 + 1(3-2) = 2.$$

Так как $z \geqslant 1$, то переходим к **Шагу 2**.

Шаг 2.

$$J = L_> \cup \{m\} = \{7, 8\},$$
$$S = \{2, 3\}.$$

Так как $|J| = 2$, то переходим к последнему шагу.

Шаг 4. Имеем $p = 1$, $q := M = 2$, $v := z = 2$. Тогда искомое решение вычисляется по формуле (см. (6)):

$$t^* = 2 - \frac{1-2}{1+1} = \frac{5}{2}.$$

4°. О лучших и худших случаях. Как уже упоминалось выше любая итерация алгоритма Maculan – de Paula состоит из Шага 1, за которым следует Шаг 2 или Шаг 3 (см. схему алгоритма рис. 2). Оценим, при каких входных данных (числовом множестве $\{c_1, \ldots, c_n\}$ и положительной константе β) алгоритм выполняет максимальное, а при каких — минимальное число итераций.

Легко показать, что для любого $n \geqslant 1$ задача (2) решается за одну итерацию в случае равенства всех элементов множества $\{c_1, \ldots, c_n\}$. Действительно, при первом прохождении Шага 1 будут образованы множества

$$L_= := 1 : n, \qquad L_> := \varnothing, \qquad L_< := \varnothing.$$

То есть, независимо от выбора следующего шага, множество J при выходе из Шага 2 или Шага 3 будет содержать только один элемент и алгоритм перейдёт к Шагу 4.

Таким образом, естественно считать случай равных элементов множества $\{c_1, \ldots, c_n\}$ лучшим случаем входных данных для алгоритма Maculan.

4.1. Во всех рассуждениях далее будем предполагать, что $c_i \neq c_j$ для всех $i \neq j$. В рамках сделанного предположения рассмотрим минимальное и максимальное значения параметра n как функции от количества итераций.

Обозначим через $d(k)$ и $D(k)$ соответственно минимальное и максимальное значение n, для которого возможны такие начальные условия, чтобы алгоритм Maculan решал задачу за k итераций.

Пусть алгоритм Maculan решает задачу (2) за $\widehat{k}$ итераций. Тогда справедливы следующие неравенства

$$d(\widehat{k}) \leqslant n \leqslant D(\widehat{k}). \tag{7}$$

В рамках сделанного предположения о неравенстве элементов входного множества $\{c_1, \ldots, c_n\}$ можно найти явные формулы для величин $d(k)$ и $D(k)$.

ТЕОРЕМА 1. *Выполняется равенство* $d(1) = 1$ *и для всех* $k \geqslant 2$ *справедлива формула*

$$d(k) = 2^{k-2} + 2. \tag{8}$$

Доказательство. Очевидно, что $d(1) = 1$. Можно убедиться, что $d(2) = 3$. Это количество шагов достигается, если во время первой итерации выполняется $z < \beta$. Далее справедливо следующее рекуррентное соотношение

$$d(k+1) = 2 \cdot (d(k) - 1), \quad k \geqslant 2. \tag{9}$$

Чтобы доказать это, рассмотрим, как изменяется мощность n входного множества $\{c_1, \ldots, c_n\}$ от итерации к итерации.

Рассмотрим входное множество $\{c_1, \ldots, c_n\}$, для которого потребуется $k+1$ итерация алгоритма Maculan. Тогда, после первой итерации алгоритма будет получено такое множество $J = \{j_1, \ldots, j_{n_1}\}$, что для множества $\{c_{j_1}, \ldots, c_{j_{n_1}}\}$ требуется k итераций алгоритма.

Если первая итерация состояла из последовательности Шаг 1+Шаг 2 (то есть на Шаге 1 было верно неравенство $z \geqslant \beta$), то $n = 2 \cdot (n_1 - 1)$ в случае четного или $n = 2 \cdot n_1 - 1$ в случае нечетного n.

Если первая итерация была Шаг 1+Шаг 3 (при $z < \beta$), то $n = 2 \cdot n_1$ или $n = 2 \cdot n_1 - 1$ соответственно.

Таким образом, наименьшее возможное значение мощности входного множества $d(k+1)$, $k \geqslant 2$, из которого после первой итерации можно получить множество мощности $d(k)$, это $2 \cdot (d(k) - 1)$ (при условии, что на первой итерации $z \geqslant \beta$). Формула (9) доказана.

Приведем полученное рекуррентное соотношение (9) к замкнутой форме. Рассмотрим функцию $Y(k) = d(k) - 2$. Тогда при $k \geqslant 2$ имеем

$$Y(k+1) + 2 = d(k+1) = 2 \cdot (d(k) - 1) = 2 \cdot (Y(k) + 2 - 1) = 2 \cdot Y(k) + 2.$$

Таким образом,

$$Y(k+1) = 2 \cdot Y(k), \quad k \geqslant 2.$$

и

$$Y(2) = d(2) - 2 = 1.$$

Следовательно,

$$Y(k) = 2^{k-2}, \quad k \geqslant 2,$$

и

$$d(k) = Y(k) + 2 = 2^{k-2} + 2, \quad k \geqslant 2.$$

Теорема доказана. □

ТЕОРЕМА 2. *Для всех $k \geqslant 1$ справедлива формула*

$$D(k) = 3 \cdot 2^{k-1}. \tag{10}$$

Доказательство. Несложно убедиться, что $D(1) = 3$. Действительно, всего одна итерация в случае $n = 3$ достигается при $z \geqslant \beta$, а при $n = 4$ требуется как минимум две итерации.

Из тех же соображений, что и в доказательстве теоремы 1 можно видеть, что максимально возможное значение для $D(k+1)$ достигается, когда после первой итерации мы получили множество J мощности $D(k)$ и при этом $D(k+1) = 2 \cdot D(k)$ (что выполняется при $z < \beta$). То есть, действует рекуррентное соотношение

$$D(k+1) = 2 \cdot D(k), \quad k \geqslant 1.$$

Несложно видеть, что в замкнутой форме это соотношение примет вид:

$$D(k) = 3 \cdot 2^{k-1}, \quad k \geqslant 1.$$

Теорема доказана. □

4.2. Теперь выразим количество итераций через функцию от n и установим взаимосвязь между введёнными величинами.

ОПРЕДЕЛЕНИЕ 2. Положим

$$k_{min}(n) := \operatorname*{argmin}_k \{D(k) \mid D(k) \geqslant n\},$$

$$k_{max}(n) := \operatorname*{argmax}_k \{d(k) \mid d(k) \leqslant n\}.$$

ТЕОРЕМА 3. *Справедливы неравенства*

$$k_{min}(n) \leqslant \widehat{k} \leqslant k_{max}(n). \tag{11}$$

Доказательство. 1) Если предположить, что $\widehat{k} < k_{min}(n)$, то по определению k_{min} и с учетом монотонного возрастания функции D имеем $D(\widehat{k}) < n$, что противоречит неравенству (7). Следовательно $k_{min}(n) \leqslant \widehat{k}$.

2) Аналогично показывается, что $\widehat{k} \leqslant k_{max}(n)$.

Теорема доказана. □

Из (11) следует, что число возможных значений для количества итераций $\widehat{k}$ равно $k_{max}(n) - k_{min}(n) + 1$.

Кроме того, из определения k_{min} и k_{max} следует, что, при условии $k_{min}(n) \geqslant 2$, выполняются неравенства

$$D(k_{min}(n) - 1) < n \leqslant D(k_{min}(n)), \tag{12}$$

$$d(k_{max}(n)) \leqslant n < d(k_{max}(n) + 1), \tag{13}$$

и, следовательно,

$$d(k_{max}(n)) \leqslant n \leqslant D(k_{min}(n)). \tag{14}$$

Таблица 1 содержит несколько первых значений d и D, которые можно посчитать по формулам (8) и (10).

k	1	2	3	4	5	6	7	8	9	10
$d(k)$	1	3	4	6	10	18	34	66	130	258
$D(k)$	3	6	12	24	48	96	192	384	768	1536

Таблица 1. Таблица значений функций $d(k)$ и $D(k)$

Для примера рассмотрим $n = 100$. Минимальное значение функции $D(k)$, которое не меньше 100, это $D(7) = 192$. Следовательно $k_{min}(n) = 7$.

В то же время, максимальное значение для $d(k)$, которое не больше 100, это $d(8) = 66$. Следовательно $k_{max}(n) = 8$.

Таким образом, любая задача при $n = 100$ потребует 7 или 8 итераций алгоритма Maculan.

4.3. Нахождение $k_{min}(n)$ и $k_{max}(n)$. Ниже нам потребуется гарантировать, что $k_{max}(n) \geqslant 2$ и $k_{min}(n) \geqslant 2$. Из таблицы 1 и определения k_{min} видно, что оба условия выполняются при $n \geqslant 4$.

Вернемся к формулам (12) и (13). Из них и полученных для функций $D(k)$ и $d(k)$ представлений (8) и (10) следует, что, при условии $n \geqslant 4$, выполняются неравенства

$$3 \cdot 2^{k_{min}(n)-2} < n \leqslant 3 \cdot 2^{k_{min}(n)-1}, \tag{15}$$

$$2^{k_{max}(n)-2} + 2 \leqslant n < 2^{k_{max}(n)-1} + 2. \tag{16}$$

Преобразовав эти формулы можно получить ограничения для значений $k_{min}(n)$ и $k_{max}(n)$ через n в явном виде. Для начала найдём выражение для $k_{min}(n)$.

ТЕОРЕМА 4. *Пусть* $n \geqslant 4$. *Тогда*

$$\log_2(n) - \log_2 3 + 1 \leqslant k_{min}(n) < \log_2(n) - \log_2 3 + 2. \tag{17}$$

Доказательство. Прологарифмируем неравенства (15) по основанию 2:

$$\log_2 3 + k_{min}(n) - 2 < \log_2(n) \leqslant \log_2 3 + k_{min}(n) - 1.$$

Вычтем из всех частей этой формулы значение $k_{min}(n)$ и домножим все на (-1), попутно разворачивая неравенства. Получим

$$1 - \log_2 3 \leqslant k_{min}(n) - \log_2(n) < 2 - \log_2 3.$$

Наконец добавим ко всем частям $\log_2(n)$. Придём к соотношениям (17).

Теорема доказана. □

СЛЕДСТВИЕ 1. *Учитывая, что* $\log_2 3 \approx 1.585$ *предыдущую оценку* (*см.* (17)) *можно переписать в виде*:

$$\log_2(n) - 0.585 < k_{min}(n) < \log_2(n) + 0.415. \tag{18}$$

СЛЕДСТВИЕ 2. *Левая и правая части в неравенстве* (17) *отличаются ровно на единицу и одно из неравенств строгое. Учитывая это и то, что* $k_{min}(n)$ *— целое число, можно сделать вывод, что* $k_{min}(n)$ *из этого выражения определяется однозначно по формуле*:

$$k_{min}(n) = \lceil \log_2(n) - \log_2 3 + 1 \rceil, \tag{19}$$

где $\lceil x \rceil = \min\{n \mid n \in \mathbb{Z}, n \geqslant x\}$ *— наименьшее большее целое для* x.

Теперь найдём оценки для $k_{max}(n)$.

ТЕОРЕМА 5. *Пусть* $n \geqslant 4$. *Справедливы неравенства*

$$\log_2(n) < k_{max}(n) < \log_2(n) + 2. \tag{20}$$

Доказательство. Прологарифмируем неравенства (16) по основанию 2:

$$k_{max}(n) - 2 + \log_2(1 + 2^{3-k_{max}(n)}) \leqslant \log_2(n) < k_{max}(n) - 1 + \log_2(1 + 2^{2-k_{max}(n)}).$$

Вычтем из всех частей неравенств значение $k_{max}(n)$ и домножим все на (-1), попутно разворачивая неравенства. Получим

$$1 - \log_2(1 + 2^{2-k_{max}(n)}) < k_{max}(n) - \log_2(n) \leqslant 2 - \log_2(1 + 2^{3-k_{max}(n)}). \tag{21}$$

Отсюда следует, что

$$k_{max}(n) = \log_2(n) + \Delta,$$

где

$$\Delta \in (\ 1 - \log_2(1 + 2^{2-k_{max}(n)})\ ,\ 2 - \log_2(1 + 2^{3-k_{max}(n)})\ ].$$

При сохранении условия $k_{max}(n) \geqslant 2$, левая граница этого интервала изменяется в пределах от 0 до $1 - \varepsilon$, а правая в пределах от примерно 0.415 до $2 - \varepsilon$, где $\varepsilon > 0$ — сколь угодно малая величина. Таким образом, упрощенная форма условий (21) будет выглядеть так

$$\log_2(n) < k_{max}(n) < \log_2(n) + 2.$$

Теорема доказана. □

Теперь объединим ограничения из выражений (18) и (20).

СЛЕДСТВИЕ 3. *Из неравенств* (11) *можно видеть, что для заданного* $n \geqslant 4$ *количество итераций алгоритма Maculan лежит в пределах*

$$\log_2(n) - 0.585 < \widehat{k} < \log_2(n) + 2. \tag{22}$$

Сформулируем основной результат этого раздела.

ТЕОРЕМА 6. *Пусть* $n \geqslant 1$. *Для любого множества* $\{c_1, \ldots, c_n\}$, *при условии, что* $c_i \neq c_j$ *для всех* $i \neq j$, *число итераций алгоритма Maculan лежит в пределах трёх возможных значений:* k, $k + 1$ *и* $k + 2$, *где*

$$k = \lceil \log_2(n) - \log_2 3 + 1 \rceil.$$

Доказательство. Прибавим неравенство $\log_2(n) - 0.585 < k_{min}(n)$ из (18) к неравенству $k_{max}(n) < \log_2(n) + 2$ из (20). Получим

$$k_{max}(n) - k_{min}(n) < \log_2(n) + 2 - \log_2(n) + 0.585 = 2.585,$$

и, поскольку разность целых чисел есть целое число, то

$$k_{max}(n) - k_{min}(n) \leqslant 2. \tag{23}$$

Таким образом, k_{max} не превосходит $k_{min} + 2$. Остается лишь добавить к этому выражение для k_{min} из формулы (19). □

Результат теоремы 6 позволяет заключить, что в случае неравенства компонент все входные данные с одинаковым n практически эквивалентны по числу итераций алгоритма. Таким образом, можно считать случай неравных компонент худшим случаем для алгоритма Maculan.

5°. Численные эксперименты. Выше отмечалось, что задача поиска ортогональной проекции точки $c = (c_1, \ldots, c_n)$ на стандартный симплекс сводится к решению уравнения (2) при β равном 1.

Были проведены массовые эксперименты проектирования точки на стандартный симплекс с использованием реализации алгоритма Maculan на языке Java. Брались 10000 точек в n-мерном евклидовом пространстве при n, равном 10, 10^2, 10^3, 10^4, 10^5 и 10^6.

Координаты точек $c = (c_1, \ldots, c_n)$ формировались тремя способами:

(A) — генерировались по непрерывному равномерному распределению на отрезке $[-10000, 10000]$;

(B) — $c_j \neq c_\ell$ для всех $1 \leqslant j \neq \ell \leqslant n$, т. е. все компоненты точки c различны;

(C) — $c_j = \xi$ для всех $1 \leqslant j \leqslant n$, где ξ — произвольное фиксированное число из отрезка $[-10000, 10000]$.

В целом численные результаты подтверждают теоретические выводы (см. рис. 3). В случаях (A) и (B) алгоритм Maculan показывает практически одинаковые результаты. Это объясняется тем, что при равномерном распределении компоненты вектора c генерируются случайным образом и их совпадение крайне маловероятно.

В случае (C) (равенства всех компонент вектора c) Maculan работает почти на порядок быстрее, чем в других случаях.

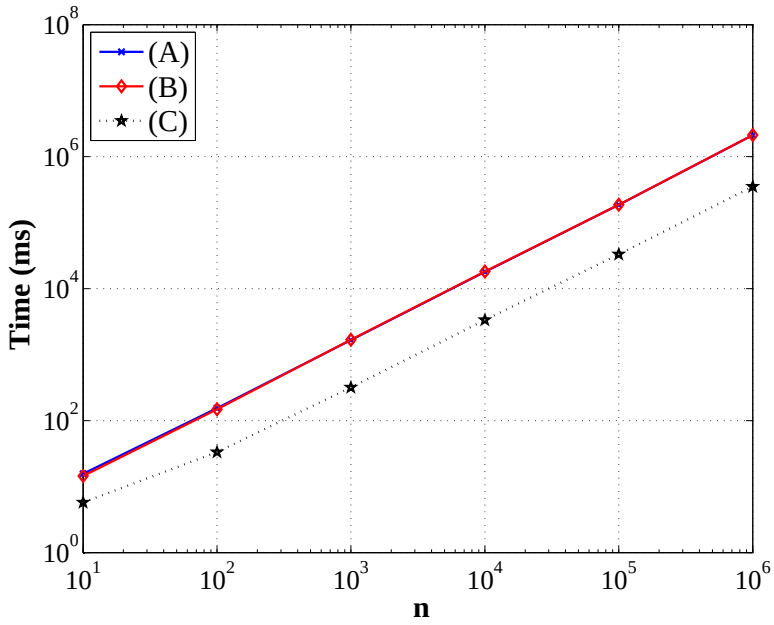

Рис. 3. Алгоритм Maculan–de Paula

ЛИТЕРАТУРА

1. Patriksson M. *A survey on the continuous nonlinear resource allocation problem* // Eur. J. Oper. Res, Vol. 185, No. 1 (Feb., 2008), pp. 1–46.

2. Held M., Wolfe P., Crowder H. P. *Validation of the subgradient optimization* // Math. Programming. 6, 1974, pp. 62–88.

3. Helgason R. V., Kennington J. L., Lall H. *A polynomially bounded algorithm for a singly constrained quadratic program* // Mathematical Programming 18, 1980, pp. 338–343.

4. Brucker P. *An $O(n)$ algorithm for quadratic knapsack problems* // Operations Research Letters 3, 1984, pp. 163–166.

5. Maculan, N., de Paula Jr., G.G. *A linear-time median-finding algorithm for projecting a vector on the simplex of* $\mathbb{R}^n$ // Operations Research Letters, 8 (4), 1989, pp. 219–222.

6. Малоземов В. Н., Певный А. Б. *Быстрый алгоритм проектирования точки на симплекс* // Вестник СПбГУ. Сер. 1. 1992. Вып. 1 (No 1). С. 112–113.

7. Causa A., Raciti F. *A purely geometric approach to the problem of computing the projection of a point on a simplex* // JOTA. 2013. Vol. 156. No 2. pp. 524–528.

8. Малоземов В. Н., Тамасян Г. Ш. *Ещё один быстрый алгоритм проектирование точки на стандартный симплекс* // Семинар «DHA & CAGD». Избранные доклады. 5 сентября 2013 г. (`http://dha.spb.ru/reps13.shtml#0905`)

9. Ахо А. *Структуры данных и алгоритмы.* Издетельский дом “Вильямс”, 2003. 382 с.