

СРАВНИТЕЛЬНОЕ ИЗУЧЕНИЕ ДВУХ БЫСТРЫХ АЛГОРИТМОВ ПРОЕКЦИРОВАНИЯ ТОЧКИ НА СТАНДАРТНЫЙ СИМПЛЕКС *)

Г. Ш. Тамасян¹, Е. В. Просолупов¹, Т. А. Ангелов¹

¹Санкт-Петербургский гос. университет,

Университетский пр., 35, 198504 Петергоф, Россия

e-mail: g.tamasyan@spbu.ru, e.prosolupov@spbu.ru, t.angelov@spbu.ru

Аннотация. Рассматриваются два алгоритма ортогонального проектирования точки на стандартный симплекс. Алгоритмы принципиально различны по своей природе, однако их связывает тот факт, что когда один из них имеет максимальную трудоёмкость, у другого трудоёмкость минимальна. Приводятся конкретные области, точки из которых проектируются рассматриваемыми алгоритмами за минимальное и максимальное число шагов. Корректность полученных выводов подтверждается численными экспериментами, реализованными в среде MatLab и независимо на языке Java. Ил. 11, библиогр. 23.

Ключевые слова: квадратичное программирование, проектирование точки на симплекс, условия оптимальности.

Введение

Задачей поиска проекции точки на множество занимаются достаточно давно, и на эту тему имеются многочисленные работы. Актуальность решения подобной задачи вызвана тем, что она встречается довольно часто, как в теории, так и на практике [21].

Следует отметить, что во многих алгоритмах (гладкой и негладкой) оптимизации [2–4, 7–9, 13–16, 22] одной из основных задач, которую приходится многократно решать, является проекция точки на замкнутое выпуклое множество [10]. Нетривиальную задачу представляет собой даже

*) Исследование выполнено при финансовой поддержке гранта СПбГУ 9.38.205.2014 и Российского фонда фундаментальных исследований (проект № 14–01–31521_мол_а).

случай, когда приходится искать проекцию на симплекс. Однако в случае, когда ищется проекция точки на *стандартный* симплекс, удаётся построить эффективные и быстрые алгоритмы.

В данной работе анализируются с точки зрения трудоёмкости два алгоритма ортогонального проецирования точки на стандартный симплекс. Впервые решение данной задачи приведено в [17], но только на идейном уровне (без обоснования). По сути исходная задача сводится к решению одного скалярного уравнения. Позднее в [5, 11, 18, 19, 23] предлагались различные численные методы, позволяющие находить решение этого уравнения.

На другом принципе основан алгоритм, предложенный Мишло [20] и имеющий наглядную геометрическую интерпретацию, что отмечено в [6, 12].

В рамках этой статьи остановимся на исследовании алгоритмов Малозёмова — Певного [5] и Мишло [20]. Следуя [6] алгоритм Малозёмова — Певного будем называть скалярным, а алгоритм Мишло — векторным.

Для рассматриваемых алгоритмов выявлены конкретные области, точки из которых проецируются за минимальное и максимальное число шагов. Хотя алгоритмы принципиально отличаются друг от друга, их связывает один интересный факт: когда один имеет максимальную трудоёмкость, у другого она минимальна.

Статья организована следующим образом. В разд. 1–3 дана формальная постановка задачи и описание алгоритмов. Разд. 4 посвящён анализу трудоёмкости алгоритмов, разд. 5 содержит результаты численных экспериментов.

1. Формулировка задачи

Рассмотрим задачу поиска проекции точки $c = (c_1, \dots, c_n)$ на стандартный симплекс $\Lambda \subset \mathbb{R}^n$, где

$$\Lambda = \left\{ x \in \mathbb{R}^n \mid \sum_{i=1}^n x_i = 1, x_i \geq 0, i = 1, \dots, n \right\}.$$

Задача формализуется так:

$$Q(x) := \frac{1}{2} \sum_{i=1}^n (x_i - c_i)^2 \longrightarrow \min_{x \in \Lambda}. \quad (1)$$

Решение этой задачи существует и единственно [1]. Обозначим его через x^* .

2. Векторный алгоритм

Начнём с описания алгоритма, идея которого предложена в [20]. Назовём его *векторным алгоритмом*.

Пусть $N = \{1, \dots, n\}$.

ПРЕДВАРИТЕЛЬНЫЙ ШАГ. В качестве начального приближения возьмём вектор $x^{(0)}$ с компонентами

$$x_i^{(0)} = c_i + \lambda, \quad i \in N, \quad (2)$$

где

$$\lambda = \frac{1}{n} \left(1 - \sum_{i \in N} c_i \right). \quad (3)$$

ОБЩИЙ ШАГ. Пусть $x^{(k)}$ — k -е приближение. Если все компоненты вектора $x^{(k)}$ неотрицательны, т. е. $x^{(k)} \geq 0$, то $x^{(k)}$ — искомая проекция точки c на стандартный симплекс Λ . Процесс заканчивается.

В противном случае, когда у вектора $x^{(k)}$ существует хотя бы одна отрицательная компонента, формируем индексное множество

$$I_k = \{i \in N \mid x_i^{(k)} \leq 0\} \quad (4)$$

и вычисляем

$$\lambda^{(k)} = \frac{1}{n_k} \left(1 - \sum_{i \in N \setminus I_k} x_i^{(k)} \right), \quad (5)$$

где $n_k = |N \setminus I_k|$. В качестве очередного приближения берём вектор $x^{(k+1)}$ с компонентами

$$x_i^{(k+1)} = \begin{cases} 0 & \text{при } i \in I_k, \\ x_i^{(k)} + \lambda^{(k)} & \text{при } i \in N \setminus I_k. \end{cases} \quad (6)$$

После этого возвращаемся к общему шагу.

Замечание. Несложно убедиться в том, что $\lambda^{(k)}$ может быть вычислено по формуле

$$\lambda^{(k)} = \frac{1}{n_k} \sum_{i \in I_k} x_i^{(k)} \quad (7)$$

и, как следствие, принимает только отрицательное значение.

3. Скалярный алгоритм

В [5] рассматривался другой алгоритм проецирования точки $c = (c_1, \dots, c_n)$ на стандартный симплекс. Назовём его *скалярным*. Напомним его описание.

ШАГ 1. Меняем знаки у компонент c_j точки c и числа $\{-c_j\}$ упорядочиваем по неубыванию. Получаем последовательность $a_1 \leq \dots \leq a_n$.

ШАГ 2. Проводим вычисления по рекуррентной формуле:

$$\begin{aligned} \varphi_1 &= 0, \\ \varphi_{k+1} &= \varphi_k + k(a_{k+1} - a_k), \quad k = 1, \dots, n-1, \end{aligned} \quad (8)$$

пока не встретим индекс k_0 , на котором $\varphi_{k_0} < 1 \leq \varphi_{k_0+1}$. Если и $\varphi_n < 1$, то полагаем $k_0 = n$.

ШАГ 3. Вычисляем λ^* по формуле

$$\lambda^* = a_{k_0} + \frac{1}{k_0}(1 - \varphi_{k_0}). \quad (9)$$

Компоненты x_i^* проекции точки c на стандартный симплекс имеют вид

$$x_i^* = (\lambda^* + c_i)_+, \quad i = 1, \dots, n, \quad (10)$$

где $(u)_+ = \max\{0, u\}$.

4. Трудоёмкость алгоритмов

С точки зрения теоретической оценки трудоёмкости скалярный алгоритм выглядит предпочтительнее, поскольку в его основе лежит рекуррентное соотношение (8) для скалярных величин, в то время как в основе векторного алгоритма — рекуррентное соотношение (6) для векторных величин.

Максимальная трудоёмкость векторного алгоритма достигается тогда, когда у всех членов последовательности $x^{(0)}, x^{(1)}, \dots$ имеется только по одной отрицательной компоненте. При этом $x^{(n-1)}$ совпадает с одним из ортов пространства \mathbb{R}^n . В общем случае векторный алгоритм требует не более чем $n^2 + 2n - 1$ арифметических операций.

Максимальная трудоёмкость скалярного алгоритма достигается тогда, когда $\varphi_{n-1} < 1$, а минимальная — при $\varphi_2 \geq 1$. (8). В общем случае скалярный алгоритм требует одну сортировку элементов массива длины n и не более чем $4n - 2$ арифметических операций.

При рассмотрении примеров в [6] замечено, что когда один алгоритм проецирования имеет максимальную трудоёмкость, у другого алгоритма трудоёмкость минимальна. Укажем конкретные области, точки из которых проецируются рассматриваемыми алгоритмами за минимальное и максимальное число шагов.

4.1. Трудоёмкость скалярного алгоритма. Введём функции

$$L_k(x) = nx_k - \sum_{i \in N} x_i + 1, \quad k \in N. \quad (11)$$

Нетрудно убедиться в том, что система неравенств $L_k(x) \geq 0$, $k \in N$, задаёт прямую призму, где одно из перпендикулярных сечений образует стандартный симплекс. В частности, при $n = 2$ данная система неравенств задаёт полосу (рис. 1), а при $n = 3$ — прямую призму (рис. 2). На рисунках стандартный симплекс выделен пунктирной линией.

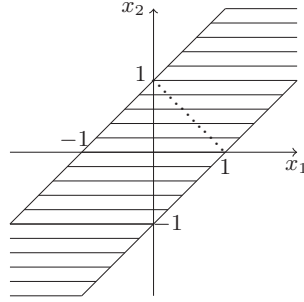


Рис. 1. Случай $n = 2$

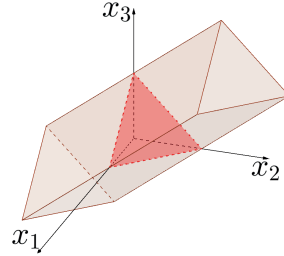


Рис. 2. Случай $n = 3$

Пусть $\lambda = \frac{1}{n} \left(1 - \sum_{i \in N} c_i\right)$.

Теорема 1. Для того чтобы проекцией точки c на стандартный симплекс Λ была точка x^* с компонентами $x_k^* = c_k + \lambda$, $k \in N$, необходимо и достаточно, чтобы $L_k(c) \geq 0$ для всех $k \in N$.

ДОКАЗАТЕЛЬСТВО. НЕОБХОДИМОСТЬ. Пусть x^* — искомая проекция точки c на Λ . Тогда $x_k^* = c_k + \lambda \geq 0$ для всех $k \in N$, т. е.

$$c_k + \frac{1}{n} \left(1 - \sum_{i \in N} c_i\right) \geq 0, \quad k \in N.$$

Если умножить на n обе части последних неравенств, то получим

$$L_k(c) = nc_k - \sum_{i \in N} c_i + 1 \geq 0 \quad \text{для всех } k \in N.$$

ДОСТАТОЧНОСТЬ. Запишем критерий оптимальности для задачи (1) [1, с. 91]:

$$\begin{aligned} x_i - c_i &= \lambda + u_i, \quad i \in N, \\ u_i x_i &= 0, \quad u_i \geq 0, \quad x_i \geq 0, \quad i \in N, \\ x_1 + \dots + x_n &= 1. \end{aligned} \quad (12)$$

Нетрудно проверить, что условия (12) выполняются при $x_i = c_i + \lambda$, $\lambda = \frac{1}{n}(1 - \sum_{i \in N} c_i)$, $u_i \equiv 0$, $i \in N$. В частности, неравенство $L_i(c) = n(c_i + \lambda) \geq 0$ справедливо при всех $i \in N$ и обеспечивает выполнение условия $x_i \geq 0$, $i \in N$. Теорема 1 доказана.

Введём множества

$$M_k = \{x \in \mathbb{R}^n \mid x_k - x_i \geq 1, \quad i \in N \setminus \{k\}\}, \quad k \in N.$$

Обозначим через $\{e_k\}$, $k \in N$, канонический базис в \mathbb{R}^n . На рис. 3 вновь пунктирной линией выделен стандартный симплекс, а множества M_1 и M_2 образуют полуплоскости. В трёхмерном случае множество

$$\{x \in \mathbb{R}^n \mid L_k(x) \geq 0, \quad k \in N\}$$

является правильной треугольной призмой (рис. 2), а M_k — конусом с вершиной в точке e_k . На рис. 4 изображено перпендикулярное сечение множеств M_k , проходящее через стандартный симплекс.

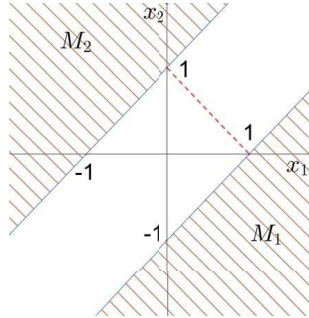


Рис. 3. Случай $n = 2$

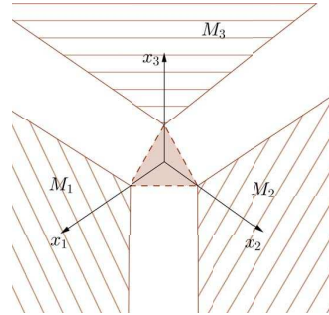


Рис. 4. Случай $n = 3$

Теорема 2. Для того чтобы проекцией точки s на стандартный симплекс Λ была точка $x^* = e_k$, необходимо и достаточно, чтобы $s \in M_k$.

ДОКАЗАТЕЛЬСТВО. ДОСТАТОЧНОСТЬ. Пусть $s \in M_k$. Покажем, что $x^* = e_k$ — проекция точки s на Λ .

Вновь воспользуемся критерием оптимальности (12). Нетрудно проверить, что эти условия выполняются при $\lambda = 1 - c_k$, $x = e_k$, $u_k = 0$, $u_i = -\lambda - c_i$, $i \in N \setminus \{k\}$. Осталось показать, что $u_i = c_k - c_i - 1 \geq 0$ для всех $i \in N \setminus \{k\}$. Действительно, согласно условию $c \in M_k$ получим $c_k - c_i \geq 1$ для всех $i \in N \setminus \{k\}$.

НЕОБХОДИМОСТЬ. Пусть x^* — искомая проекция точки c на Λ . Покажем, что $c \in M_k$.

Условия оптимальности (12) выполняются при $\lambda = 1 - c_k$, $x^* = e_k$, $u_k = 0$ и $u_i \geq 0$, $i \in N \setminus \{k\}$.

Рассмотрим равенства в первой строке условий (12) при $i \in N \setminus \{k\}$. Получим

$$u_i = -\lambda - c_i, \quad i \in N \setminus \{k\}.$$

На основании того, что $u_i \geq 0$ при всех $i \in N \setminus \{k\}$ и $\lambda = 1 - c_k$, приходим к соотношениям

$$u_i = c_k - c_i - 1 \geq 0, \quad i \in N \setminus \{k\}.$$

Выполнение этих неравенств и означает, что $c \in M_k$. Теорема 2 доказана.

Теорема 3. *Максимальная трудоёмкость скалярного алгоритма достигается тогда и только тогда, когда $L_k(c) \geq 0$ при всех $k \in N$.*

ДОКАЗАТЕЛЬСТВО. Согласно скалярному алгоритму имеем

$$\begin{aligned} \varphi_1 &= 0, & \varphi_2 &= -a_1 + a_2, & \varphi_3 &= -a_1 - a_2 + 2a_3, \\ \varphi_n &= -a_1 - a_2 - \dots - a_{n-1} + (n-1)a_n. \end{aligned}$$

НЕОБХОДИМОСТЬ. Пусть $\varphi_n \leq 1$. Покажем, что $L_k(c) \geq 0$ при всех $k \in N$. Действительно, если вместо $\{a_i\}$ в неравенство $\varphi_n \leq 1$ подставить $\{-c_j\}$, получим

$$c_{k_1} + c_{k_2} + \dots + c_{k_{n-1}} - (n-1)c_{k_n} \leq 1, \quad (13)$$

где k_1, \dots, k_n — некоторая перестановка чисел $1, \dots, n$. Перепишем (13) в виде

$$nc_{k_n} - \sum_{j \in N} c_{k_j} + 1 \geq 0,$$

т. е. $L_k(c) \geq 0$ при всех $k \in N$.

ДОСТАТОЧНОСТЬ. Пусть $L_k(c) \geq 0$ при всех $k \in N$. Покажем, что $\varphi_n \leq 1$. Запишем систему неравенств $L_k(c) \geq 0$ при всех $k \in N$ в виде

$$nc_{k_n} - \sum_{j \in N} c_{k_j} + 1 \geq 0, \quad (14)$$

где k_1, \dots, k_n — некоторая перестановка чисел $1, \dots, n$, а числа $\{-c_{k_j}\}$ упорядочены по неубыванию. Далее, согласно алгоритму строим последовательность $a_1 \leq \dots \leq a_n$. Тогда неравенство (14) примет вид

$$-na_n + \sum_{j \in N} a_j + 1 \geq 0,$$

что эквивалентно тому, что $\varphi_n \leq 1$. Теорема 3 доказана.

Теорема 4. Минимальная трудоёмкость скалярного алгоритма достигается тогда и только тогда, когда $c \in M_k$ при некотором $k \in N$.

ДОКАЗАТЕЛЬСТВО. ДОСТАТОЧНОСТЬ. Пусть $c \in M_k$, т. е.

$$-c_k \leq -c_i - 1, \quad i \in N \setminus \{k\}. \quad (15)$$

В силу (15) на первом шаге скалярного алгоритма получим $a_1 = -c_k$. Далее вычислим $\varphi_1 = 0$ и $\varphi_2 = a_2 - a_1$, где a_2 может принять любое значение $\{-c_i\}$, $i \in N \setminus \{k\}$. Согласно условиям теоремы и (15) заключаем, что

$$\varphi_2 = a_2 - a_1 = c_k - c_i \geq 1, \quad i \in N \setminus \{k\}.$$

Таким образом, вычисления $\{\varphi_k\}$ закончились на первом шаге.

НЕОБХОДИМОСТЬ. Пусть трудоёмкость скалярного алгоритма минимальна, т. е.

$$\varphi_2 = a_2 - a_1 \geq 1. \quad (16)$$

Здесь $a_1 := \min_{j \in N} \{-c_j\} = -c_k$ при некотором $k \in N$ и $a_2 := -c_i$ при некотором $i \in N \setminus \{k\}$. Подставив $a_1 = -c_k$ и $a_2 = -c_i$ в (16), получим (15), т. е. $c \in M_k$. Теорема 4 доказана.

4.2. Трудоёмкость векторного алгоритма.

Теорема 5. Для того чтобы трудоёмкость векторного алгоритма была минимальной, необходимо и достаточно, чтобы $L_k(c) \geq 0$ при всех $k \in N$.

ДОКАЗАТЕЛЬСТВО немедленно следует из теоремы 1 и предварительного шага векторного алгоритма (см. (2), (3)).

Следствие 1. Из теорем 1, 3 и 5 следует, что если при проецировании точки c на стандартный симплекс Λ у скалярного алгоритма достигается максимальная трудоёмкость, то для векторного алгоритма трудоёмкость минимальна, а именно, векторный алгоритм закончится на предварительном шаге.

Пусть ℓ_1, \dots, ℓ_n — некоторая перестановка чисел $\{1, \dots, n\}$.

Теорема 6. Трудоемкость векторного алгоритма максимальна тогда и только тогда, когда найдётся такая перестановка ℓ_1, \dots, ℓ_n , при которой компоненты точки s удовлетворяют следующим неравенствам:

$$c_{\ell_{j-1}} < c_{\ell_j} + \left[(n-j)c_{\ell_j} - \sum_{k=j+1}^n c_{\ell_k} + 1 \right], \quad j = 2, \dots, n-1, \quad (17)$$

$$c_{\ell_{n-1}} < c_{\ell_n} - 1.$$

Доказательству этой теоремы предпшлём несколько вспомогательных утверждений.

Для простоты изложения будем считать, что $\ell_j = j$, $j \in N$. Предположим, что компоненты точки $s \in \mathbb{R}^n$ удовлетворяют условиям (17). Тогда неравенства (17) примут вид

$$\begin{aligned} c_{n-1} &< c_n - 1, \\ c_{n-2} &< c_{n-1} + [c_{n-1} - c_n + 1], \\ c_{n-3} &< c_{n-2} + [2c_{n-2} - c_{n-1} - c_n + 1], \\ &\dots \\ c_2 &< c_3 + [(n-3)c_3 - \sum_{k=4}^n c_k + 1], \\ c_1 &< c_2 + [(n-2)c_2 - \sum_{k=3}^n c_k + 1]. \end{aligned} \quad (18)$$

Лемма 1. Если компоненты точки $s \in \mathbb{R}^n$ удовлетворяют условиям (18), то

$$c_1 < c_2 < \dots < c_{n-1} < c_n - 1. \quad (19)$$

Значит, точка s принадлежит множеству $\text{int } M_n$ (см. (15)).

Доказательство. Достаточно показать, что выражения в квадратных скобках в правых частях неравенств (18) отрицательны.

Действительно, выражение в квадратных скобках во втором неравенстве меньше нуля в силу первого неравенства. Тогда $c_{n-2} < c_{n-1}$, а следовательно, $c_{n-2} < c_n - 1$. Тем самым в третьем неравенстве выражение в квадратных скобках меньше нуля. Аналогично доказываются и оставшиеся неравенства.

Таким образом, можно показать, что $c_n - c_k > 1$ для всех $k = 1, \dots, n-1$, откуда и следует, что точка s принадлежит множеству $\text{int } M_n$. Лемма 1 доказана.

Следствие 2. Пусть W — множество точек c из \mathbb{R}^n , удовлетворяющих условиям (17). Тогда $W \subset \text{int } M_{\ell_n}$.

Лемма 2. Если компоненты точки $c \in \mathbb{R}^n$ удовлетворяют условиям (18), то

$$\begin{cases} L_1(c) = nc_1 - \sum_{j \in N} c_j + 1 < 0, \\ L_2(c) = nc_2 - \sum_{j \in N} c_j + 1 > 0, \\ \dots \\ L_{n-1}(c) = nc_{n-1} - \sum_{j \in N} c_j + 1 > 0, \\ L_n(c) = nc_n - \sum_{j \in N} c_j + 1 > n. \end{cases} \quad (20)$$

Доказательство. Справедливость первого и последнего неравенств следует из (19). Покажем, что $L_k(c) > 0$ для всех $k = 2, \dots, n-1$. Перепишем выражение для $L_k(c)$ в следующем виде:

$$\begin{aligned} L_k(c) &= nc_k - \sum_{j \in N} c_j + 1 \\ &= \left(c_k - c_{k-1} + \left[(n-k)c_k - \sum_{j=k+1}^n c_j + 1 \right] \right) + \left(kc_k - \sum_{j=1}^k c_j \right) - c_k + c_{k-1} \\ &= \left(c_k - c_{k-1} + \left[(n-k)c_k - \sum_{j=k+1}^n c_j + 1 \right] \right) + \sum_{j=1}^{k-2} (c_k - c_j). \end{aligned}$$

Первая группа слагаемых положительна в силу k -го неравенства в (18), а положительность второго слагаемого следует из (19). Лемма 2 доказана.

Лемма 3. Если компоненты точки $c \in \mathbb{R}^n$ удовлетворяют условиям (18), то

$$\sum_{j=1}^{k-1} c_j - (k-1)c_k + L_k(c) < 0 \quad \text{для всех } k = 2, \dots, n-1, \quad (21)$$

$$\sum_{j=1}^{k-1} c_j - (k-1)c_\ell + L_\ell(c) > 0 \quad \text{при } k+1 \leq \ell \leq n-1, \quad (22)$$

$$\sum_{j=1}^{k-1} c_j - (k-1)c_n + L_n(c) > n - k + 1. \quad (23)$$

ДОКАЗАТЕЛЬСТВО. Вначале рассмотрим неравенства (21). С учётом (11) преобразуем левую часть (21):

$$\begin{aligned} \sum_{j=1}^{k-1} c_j - (k-1)c_k + L_k(c) &= \sum_{j=1}^{k-1} c_j - (k-1)c_k + nc_k - \sum_{j \in N} c_j + 1 \\ &= (n-k)c_k - \sum_{j=k+1}^n c_j + 1 = \sum_{j=k+1}^{n-1} (c_k - c_j) + (c_k - c_n + 1). \end{aligned}$$

Отсюда в силу (19) следует справедливость неравенств (21).

Докажем неравенства (22). Используя то, что для допустимых ℓ выполнено неравенство $c_\ell \geq c_{k+1}$ (см. (19)), преобразуем левую часть (22) к виду

$$\begin{aligned} \sum_{j=1}^{k-1} c_j - (k-1)c_\ell + L_\ell(c) &= c_\ell - c_k + \sum_{j=k+1}^n (c_\ell - c_j) + 1 \\ &\geq c_{k+1} - c_k + \sum_{j=k+2}^n (c_{k+1} - c_j) + 1 > 0. \end{aligned}$$

Последнее неравенство является следствием одного из неравенств (18).

Справедливость (23) следует из того, что неравенство $c_n - c_j > 1$ выполнено для всех $j = 1, \dots, n-1$. Действительно, рассмотрим и преобразуем левую часть (23):

$$\begin{aligned} \sum_{j=1}^{k-1} c_j - (k-1)c_n + L_n(c) &= \sum_{j=1}^{k-1} c_j - (k-1)c_n + nc_n - \sum_{j \in N} c_j + 1 \\ &= \sum_{j=k}^{n-1} (c_n - c_j) + 1 > n - k + 1. \end{aligned}$$

Лемма 3 доказана.

ДОКАЗАТЕЛЬСТВО ТЕОРЕМЫ 6. НЕОБХОДИМОСТЬ. Пусть трудоёмкость векторного алгоритма максимальна. Это означает, что индексные множества I_k (см. (4)) на каждом шаге пополнялись только одним элементом. Не умаляя общности, предположим, что $I_0 = \{1\}$, $I_k = I_{k-1} \cup \{k+1\}$, $k = 1, \dots, n-2$. Нетрудно заметить, что в этом случае $\ell_n = n$. И вновь для простоты изложения будем считать, что $\ell_j = j$,

$j \in N$. Покажем, что тогда точка c удовлетворяет неравенствам (18) и согласно теореме 2 и лемме 1 проекцией точки c на стандартный симплекс Λ будет точка $x^* = e_n$.

На предварительном шаге векторного алгоритма находим величину $\lambda = \frac{1}{n} \left(1 - \sum_{j \in N} c_j\right)$ и вектор $x^{(0)}$ с компонентами

$$x_i^{(0)} = c_i + \lambda = \frac{1}{n} \left(n c_i - \sum_{j \in N} c_j + 1 \right), \quad i \in N.$$

По предположению на первом шаге имеем $I_0 = \{1\}$, тогда (см. (11))

$$\begin{cases} x_1^{(0)} = \frac{1}{n} L_1(c) < 0, \\ x_2^{(0)} = \frac{1}{n} L_2(c) > 0, \\ \dots, \\ x_{n-1}^{(0)} = \frac{1}{n} L_{n-1}(c) > 0, \\ x_n^{(0)} = \frac{1}{n} L_n(c) > 0. \end{cases}$$

Заметим, что если в последнем неравенстве в (18) c_1 перенести в правую часть, то получим $L_2(c)$, которое положительно. Таким образом, справедливость последнего неравенства в (18) показана.

Далее находим (см. (5)–(7))

$$\lambda^{(0)} = \frac{1}{n-1} x_1^{(0)} = \frac{1}{n(n-1)} L_1(c).$$

Первое приближение принимает вид

$$x^{(1)} = \begin{pmatrix} 0 \\ x_2^{(0)} + \lambda^{(0)} \\ \dots \\ x_n^{(0)} + \lambda^{(0)} \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{(n-1)} (c_1 - c_2 + L_2(c)) \\ \dots \\ \frac{1}{(n-1)} (c_1 - c_n + L_n(c)) \end{pmatrix}.$$

На втором шаге индексное множество I_1 состоит из двух элементов $\{1, 2\}$, т. е.

$$\begin{cases} x_1^{(1)} = 0, \\ x_2^{(1)} = \frac{1}{(n-1)} (c_1 - c_2 + L_2(c)) < 0, \\ x_3^{(1)} = \frac{1}{(n-1)} (c_1 - c_3 + L_3(c)) > 0, \\ \dots \\ x_{n-1}^{(1)} = \frac{1}{(n-1)} (c_1 - c_{n-1} + L_{n-1}(c)) > 0, \\ x_n^{(1)} = \frac{1}{(n-1)} (c_1 - c_n + L_n(c)) > 0. \end{cases}$$

Если в предпоследнем неравенстве в (18) c_2 перенести в правую часть, то получим выражение $c_1 - c_3 + L_3(c)$, которое положительно, так как $x_3^{(1)} > 0$. Таким образом, справедливость предпоследнего неравенства в (18) также показана.

Вычислим

$$\lambda^{(1)} = \frac{1}{n-2} x_2^{(1)} = \frac{1}{(n-1)(n-2)} (c_1 - c_2 + L_2(c)).$$

Второе приближение примет вид

$$x^{(2)} = \begin{pmatrix} 0 \\ 0 \\ x_3^{(1)} + \lambda^{(1)} \\ \dots \\ x_n^{(1)} + \lambda^{(1)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{(n-2)} (c_1 + c_2 - 2c_3 + L_3(c)) \\ \dots \\ \frac{1}{(n-2)} (c_1 + c_2 - 2c_n + L_n(c)) \end{pmatrix}.$$

Наконец, на последнем шаге имеем $I_{n-2} = \{1, \dots, n-1\}$. Тогда

$$\begin{cases} x_1^{(n-2)} = \dots = x_{n-2}^{(n-2)} = 0, \\ x_{n-1}^{(n-2)} = \frac{1}{2} \left(\sum_{i=1}^{n-2} c_i - (n-2)c_{n-1} + L_{n-1}(c) \right) < 0, \\ x_n^{(n-2)} = \frac{1}{2} \left(\sum_{i=1}^{n-2} c_i - (n-2)c_n + L_n(c) \right) > 1. \end{cases} \quad (24)$$

Преобразовав последнее неравенство, получим $c_n - c_{n-1} - 1 > 0$. Таким образом, справедливость первого неравенства в (18) доказана.

Теперь вычислим

$$\lambda^{(n-2)} = \frac{1}{2} \left(\sum_{i=1}^{n-2} c_i - (n-2)c_{n-1} + L_{n-1}(c) \right).$$

Отсюда в силу замечания, что все $\lambda^{(k)}$ принимают только отрицательные значения, получим неравенство

$$\lambda^{(n-2)} = c_{n-1} + 1 - c_n < 0. \quad (25)$$

В итоге последнее приближение примет вид

$$x^{(n-1)} = e_n = x^*,$$

так как $x_i^{(n-1)} = 0$ при $i \in 1, \dots, n-1$ и $x_n^{(n-1)} := x_n^{(n-2)} + \lambda^{(n-2)} = 1$.

Аналогично можно рассмотреть иные закономерности образования индексных множеств I_k , $k = 1, \dots, n-2$, из элементов $\{1, \dots, n-1\}$ и получить следующие неравенства (см. (25)):

$$c_i + 1 - c_n < 0, \quad i = 1, \dots, n-1, \quad (26)$$

тем самым точка c принадлежит множеству $\text{int } M_n$.

ДОСТАТОЧНОСТЬ. Пусть компоненты точки c удовлетворяют неравенствам (18). По лемме 2 имеем $I_0 = \{1\}$. Следуя алгоритму, описанному выше, можно показать, что в силу леммы 3 индексные множества I_k на каждом шаге пополняются только одним элементом, а именно,

$$I_k = I_{k-1} \cup \{k+1\}, \quad k = 1, \dots, n-2.$$

Таким образом, показана максимальная трудоёмкость векторного алгоритма. Теорема 6 доказана.

Из теорем 2, 4, 6 и следствия 2 вытекает

Следствие 3. Если при проецировании точки c на стандартный симплекс Λ у векторного алгоритма достигается максимальная трудоёмкость, то для скалярного алгоритма трудоёмкость минимальна.

4.3. Сложность практической реализации случая с максимальной трудоёмкостью векторного алгоритма. Далее покажем, что существуют препятствия для практической генерации точки $c \in \mathbb{R}^n$, удовлетворяющей системе (18). Обозначим

$$\xi_1 = c_n - c_{n-1} - 1, \quad \xi_j = c_{n-j+1} - c_{n-j} \quad \text{для всех } j = 2, \dots, n-1. \quad (27)$$

Тогда

$$c_n - c_1 = (c_n - c_{n-1}) + (c_{n-1} - c_{n-2}) + \dots + (c_2 - c_1) = 1 + \sum_{j=1}^{n-1} \xi_j.$$

Попробуем выяснить, какова минимальная возможная разность между c_n и c_1 для точки, удовлетворяющей системе (18). Преобразуем выражение в квадратных скобках в правой части k -го неравенства в (18). В силу (27) справедливо

$$1 + (n-k-1)c_{k+1} - \sum_{j=k+2}^n c_j = - \sum_{j=1}^{n-k-1} j\xi_j.$$

Следовательно, система (18) примет вид

$$\begin{cases} c_{n-1} < c_n - 1, \\ c_k < c_{k+1} - \sum_{j=1}^{n-k-1} j \xi_j, \quad k = 1, \dots, n-2. \end{cases}$$

Отсюда с учётом определения величин ξ_j получим систему неравенств

$$\begin{cases} \xi_1 > 0, \\ \xi_{n-k} > \sum_{j=1}^{n-k-1} j \xi_j, \quad k = 1, \dots, n-2. \end{cases} \quad (28)$$

Обозначим через δ наименьшую доступную на вычислительном устройстве разницу между двумя числами. Тогда для каждого неравенства в системе (28) можем считать, что правая и левая части отличаются ровно на δ . Такое предположение позволит нам определить наименьшую возможную разницу между первой и последней координатами. Полученный таким образом набор минимальных разностей компонент вектора c обозначим через $\hat{\xi}_1, \dots, \hat{\xi}_{n-1}$:

$$\begin{cases} \hat{\xi}_1 = \delta, \\ \hat{\xi}_t = \delta + \sum_{j=1}^{t-1} j \hat{\xi}_j, \quad t = 2, \dots, n-1. \end{cases} \quad (29)$$

Несложно проверить, что $\hat{\xi}_t = t!\delta$, $t = 1, \dots, n-1$.

Стало быть, для точки $c \in \mathbb{R}^n$ с минимальными возможными разностями между компонентами при условии выполнения неравенств (18), будет верно

$$c_n - c_1 = 1 + \sum_{j=1}^{n-1} \xi_j = 1 + \delta \sum_{j=1}^{n-1} j!. \quad (30)$$

Поскольку значение факториала растёт очень быстро, разность между c_n и c_1 будет очень большой уже при достаточно малых n , даже если значение δ для устройства будет очень мало.

Например, для $\delta = 10^{-6}$ и разности между минимальной и максимальной координатами точки c , не превосходящей 20000, получим, что максимальная размерность, для которой создание такой точки возможно, это $n = 14$.

Даже если представить устройство, для которого $\delta = 10^{-150}$, то уже при размерности $n = 170$ разброс значений компонент составил бы

$$c_n - c_1 = 4.2945 \cdot 10^{154}.$$

Эти рассуждения позволяют нам сделать вывод, что при больших размерностях худший случай для векторного алгоритма практически недостижим.

5. Численные эксперименты

Ниже представлены экспериментальные результаты поведения описанных выше алгоритмов при разном распределении проецируемых точек. Для получения практических данных исследуемые алгоритмы были реализованы на языке Java и в среде MatLab.

Измерения проводились для размерностей $10, 10^2, 10^3, 10^4, 10^5, 10^6$. Для каждой из четырёх реализаций алгоритмов проводилось по 10000 запусков для каждой из размерностей и замерялось суммарное время в наносекундах. Во всех случаях точки генерировались таким образом, чтобы значение каждой координаты лежало в пределах от -10000 до 10000 . Для лучшей читаемости результатов при подготовке графиков использовалось время в миллисекундах и логарифмический масштаб.

Далее на графиках используются следующие обозначения: Java:Scalar и Java:Vector — скалярный и векторный алгоритмы, реализованные на Java; MatLab:Scalar и MatLab:Vector — те же алгоритмы, реализованные в среде MatLab; Uniform — случай равномерно распределённых в пространстве точек; ScalarBest — лучший случай для скалярного алгоритма (теорема 4); SimplexPrism — лучший случай для векторного алгоритма (теорема 5), который является в то же время худшим для скалярного (теорема 3).

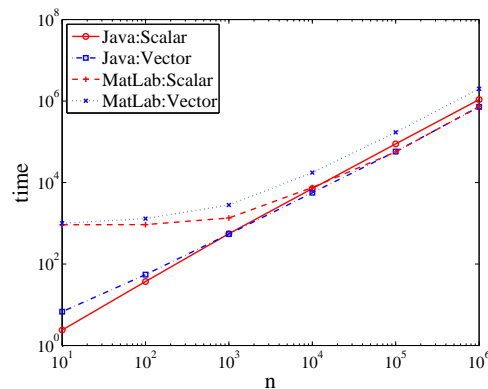


Рис. 5. Равномерно распределённые точки

Как сказано выше, сгенерировать точки для худшего случая векторного алгоритма для большинства рассматриваемых нами размерностей

не представляется практически возможным. По этой причине сравнение работы алгоритмов для этого случая не проводилось.

Экспериментальные результаты в целом показывают, что основной слабостью скалярного алгоритма является необходимость предварительной сортировки входных данных. На рис. 5 сравнивается время работы реализаций алгоритмов при равномерно распределённых в пространстве проецируемых точках. Можно видеть, что результаты сравнения алгоритмов на MatLab и Java различаются. Это вызвано тем, что функция сортировки включена в вычислительное ядро MatLab, а при реализации на Java сортировка является просто частью программы. Поэтому влияние времени сортировки на реализацию скалярного алгоритма в MatLab меньше.

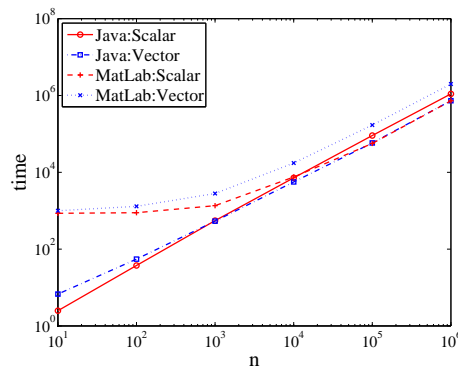


Рис. 6. ScalarBest — лучший случай скалярного алгоритма

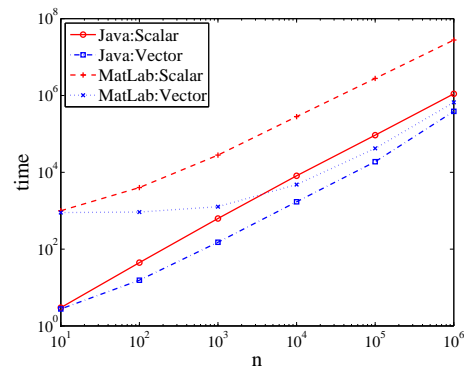


Рис. 7. SimplexPrism — лучший случай векторного алгоритма

На размерностях до 1000 лидером здесь является скалярный алгоритм, реализованный на Java. При больших размерностях (от 10^5) быстрее всего работают скалярный алгоритм в MatLab и векторный на Java. Векторный алгоритм в среде MatLab безусловно проигрывает.

На рис. 6 приведён график для наиболее благоприятного для скалярного алгоритма случая. Можно заметить, что он не отличается от графика на рис. 5. Как будет видно из следующих графиков, практически время работы скалярного алгоритма в его лучшем случае не отличается от времени при равномерном распределении.

На рис. 7 можно видеть, что в области наименьшей трудоёмкости векторного алгоритма и наибольшей трудоёмкости скалярного векторный алгоритм выигрывает во всех реализациях. При этом реализации обоих алгоритмов на Java значительно выигрывают по сравнению с MatLab. В особенности это касается реализаций скалярного алгоритма, для которого необходимость проделать все шаги практически не сказывается на

времени работы на Java, но значительно замедляет выполнение скрипта MatLab.

Рассмотрим, как каждый из алгоритмов отдельно ведёт себя в разных случаях. На рис. 8 и 9 изображено поведение скалярного алгоритма при разных способах генерации проецируемой точки. Графики подтверждают, что скорость работы скалярного алгоритма в чистом виде ограничена снизу скоростью сортировки, которая производится на его подготовительном этапе. При распределении проецируемых точек в области, наиболее благоприятной для этого алгоритма, время работы не отличается от случая точек, равномерно распределённых в пространстве. В худшем случае, на точках SimplexPrism, реализация в MatLab работает гораздо медленнее, чем на других данных. Реализация же алгоритма на языке Java полностью подчинена времени сортировки, и в худшем случае время её работы, как и в лучшем, не отличается от среднего.

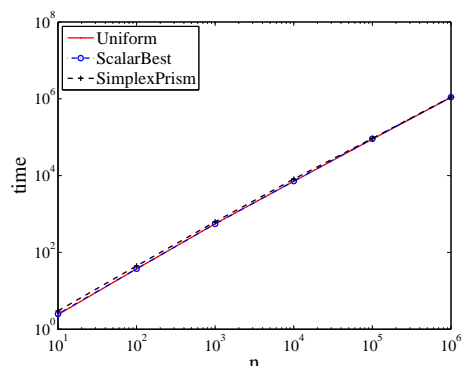


Рис. 8. Java:Scalar — скалярный алгоритм на Java

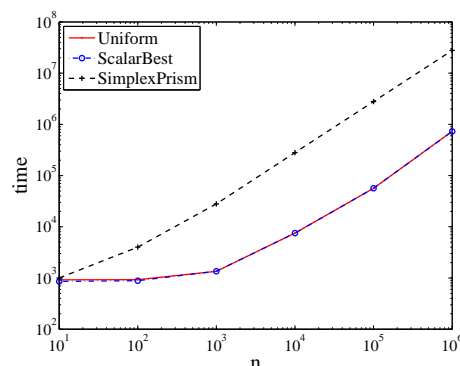


Рис. 9. MatLab:Scalar — скалярный алгоритм на MatLab

На рис. 10 и 11 представлено поведение векторного алгоритма. Эти графики подтверждают теоретическое заключение: алгоритм в обеих реализациях работает заметно быстрее в точках из призмы, трёхмерный случай которой изображён на рис. 2.

Заключение

В работе рассмотрены два алгоритма поиска ортогональной проекции точки на стандартный симплекс. Выявлены случаи наибольшей и наименьшей трудоёмкости обоих алгоритмов.

В частности, при реализации в MatLab скалярный алгоритм выигрывает у векторного на случайных точках, но проигрывает в области своей наихудшей производительности. В реализации же на Java скалярный алгоритм выигрывает в тех же случаях, но при размерностях до 1000. При

бóльших размерностях векторный алгоритм на Java показывает лучшие результаты, чем скалярный.

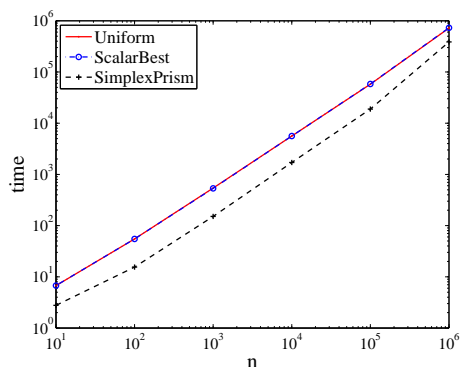


Рис. 10. Java:Vector — векторный алгоритм на Java

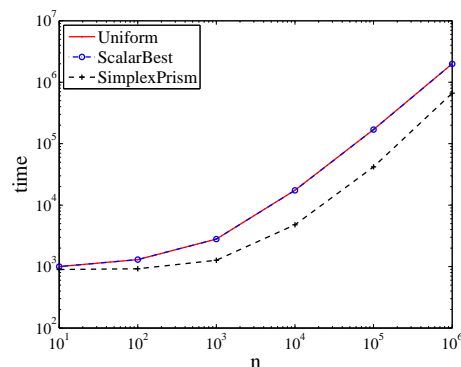


Рис. 11. MatLab:Vector — векторный алгоритм на MatLab

Время работы скалярного алгоритма определяется главным образом необходимостью сортировать элементы исходного массива. Это не позволяет скалярному алгоритму работать с наибольшей эффективностью в среднем и даже в самом благоприятном случае.

Следует отметить, что для скалярного алгоритма можно получить значительно лучшие результаты, если усовершенствовать его подготовительный этап. В [23] приведены результаты испытаний таких модификаций. Эксперименты показывают выигрыш до десятикратного при замене полной сортировки в начале на поэтапное нахождение элементов отсортированной последовательности.

ЛИТЕРАТУРА

1. Гавурин М. К., Малозёмов В. Н. Экстремальные задачи с линейными ограничениями. Л.: Изд-во ЛГУ, 1984. 176 с.
2. Демьянов В. Ф., Тамасян Г. Ш. О прямых методах решения вариационных задач // Тр. Ин-та математики и механики УрО РАН. 2010. Т. 16, № 5. С. 36–47.
3. Долгополик М. В., Тамасян Г. Ш. Об эквивалентности методов наискорейшего и гиподифференциального спусков в некоторых задачах условной оптимизации // Изв. Саратов. ун-та. Нов. сер. Сер. Математика. Механика. Информатика. 2014. Т. 14, вып. 4, ч. 2. С. 532–542.
4. Малозёмов В. Н. МДМ-методу — 40 лет // Вестн. Сыктывкар. ун-та. Сер. 1. 2012. Вып. 15. С. 51–62.
5. Малозёмов В. Н., Певный А. Б. Быстрый алгоритм проектирования точки на симплекс // Вестн. СПбГУ. Сер. 1. 1992. № 1. С. 112–113.

6. Малозёмов В. Н., Тамасян Г. Ш. Два быстрых алгоритма проектирования точки на стандартный симплекс // Журн. вычисл. математики и мат. физики. 2016. DOI: 10.7868/S0044466916050148 (в печати).
7. Тамасян Г. Ш. О методах наискорейшего и гиподифференциального спуска в одной задаче вариационного исчисления // Вычисл. методы и программирование: новые вычисл. технологии. 2012. Т. 13, № 1. С. 197–217.
8. Тамасян Г. Ш. Численные методы в задачах вариационного исчисления для функционалов, зависящих от производных высшего порядка // Пробл. мат. анализа. Вып. 67. Новосибирск: Изд-во Тамара Рожковская, 2012. С. 113–132.
9. Тамасян Г. Ш., Чумаков А. А. Нахождение расстояния между эллипсоидами // Дискрет. анализ и исслед. операций. 2014. Т. 21, № 3. С. 87–102.
10. Утешев А. Ю., Яшина М. В. Нахождение расстояния от эллипсоида до плоскости и квадрики в \mathbb{R}^n // Докл. АН. 2008. Т. 419, № 4. С. 471–474.
11. Brucker P. An $O(n)$ algorithm for quadratic knapsack problems // Oper. Res. Lett. 1984. Vol. 3, No. 3. P. 163–166.
12. Causa A., Raciti F. A purely geometric approach to the problem of computing the projection of a point on a simplex // J. Optimization Theory Appl. 2013. Vol. 156, No. 2. P. 524–528.
13. Demyanov V. F., Giannessi F., Tamasyan G. Sh. Variational control problems with constraints via exact penalization // Variational Analysis and Applications. Eds. F. Giannessi, A. Maugeru. New York: Springer-Verl., 2005. P. 301–342. (Nonconvex Optim. Its Appl.; Vol. 79).
14. Demyanov V. F., Tamasyan G. Sh. Exact penalty functions in isoperimetric problems // Optimization. 2011. Vol. 60, No. 1–2. P. 153–177.
15. Demyanov V. F., Tamasyan G. Sh. Direct methods in the parametric moving boundary variational problem // Numer. Funct. Anal. Optimization. 2014. Vol. 35, No. 7–9. P. 932–961.
16. Dolgopolik M. V., Tamasyan G. Sh. Method of steepest descent for two-dimensional problems of calculus of variations // Constructive Nonsmooth Analysis and Related Topics. (V. F. Demyanov, P. M. Pardalos, and M. Batsyn, eds.) New York: Springer-Verl., 2014. P. 101–113. (Optim. Its Appl.; Vol. 87).
17. Held M., Wolfe P., Crowder H. P. Validation of the subgradient optimization // Math. Progr. 1974. Vol. 6, No. 1. P. 62–88.
18. Helgason R. V., Kennington J. L., Lall H. A polynomially bounded algorithm for a singly constrained quadratic program // Math. Program. 1980. Vol. 18, No. 1. P. 338–343.
19. Maculan N., Galdino de Paula G., Jr. A linear-time median-finding algorithm for projecting a vector on the simplex of \mathbb{R}^n // Oper. Res. Lett. 1989. Vol. 8, No. 4. P. 219–222.

- 20. **Michelot C.** A finite algorithm for finding the projection of a point onto the canonical simplex of \mathbb{R}^n // J. Optimization Theory Appl. 1986. Vol. 50, No. 1. P. 195–200.
- 21. **Patriksson M.** A survey on the continuous nonlinear resource allocation problem // Eur. J. Oper. Res. 2008. Vol. 185, No. 1. P. 1–46.
- 22. **Tamasyan G. Sh., Chumakov A. A.** Finding the distance between the ellipsoid and the intersection of a linear manifold and ellipsoid // Proc. 2015 Int. Conf. “Stability and Control Processes” in Memory of V.I. Zubov (SCP) joined with 21st Int. Workshop on Beam Dynamics and Optimization (BDO) (St. Petersburg, Russia, Oct. 5–9, 2015). IEEE, 2015. P. 357–360. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7342138>
- 23. **Tamasyan G., Prosolupov E.** Orthogonal projection of a point onto the standard simplex algorithms analysis // Proc. 2015 Int. Conf. “Stability and Control Processes” in Memory of V.I. Zubov (SCP) joined with 21st Int. Workshop on Beam Dynamics and Optimization (BDO) (St. Petersburg, Russia, Oct. 5–9, 2015). IEEE, 2015. P. 353–356. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7342137>

*Тамасян Григорий Шаликович,
Просолупов Евгений Викторович,
Ангелов Тодор Ангелов*

Статья поступила
11 сентября 2015 г.
Исправленный вариант —
19 октября 2015 г.

DISKRETNYYI ANALIZ I ISSLEDOVANIE OPERATSII
April–June 2016. Volume 23, No. 2. P. 100–123

UDC 519.85

DOI: 10.17377/daio.2016.23.510

COMPARATIVE STUDY OF TWO FAST ALGORITHMS
FOR PROJECTING A POINT TO THE STANDARD SIMPLEX

G. Sh. Tamasyan¹, E. V. Prosolupov¹, and T. A. Angelov¹

¹St. Petersburg State University,
35 University Ave., 198504 Peterhof, Russia

e-mail: g.tamasyan@spbu.ru, e.prosolupov@spbu.ru, t.angelov@spbu.ru

Abstract. We consider two algorithms for orthogonal projection of a point to the standard simplex. Although these algorithms are fundamentally different, the following fact unites them. When one of them has the maximum run time, the run time of the other is minimal. Some particular domains are presented whose points are projected by the considered algorithms in the minimum and maximum number of iterations. The correctness of the conclusions is confirmed by the numerical experiments independently implemented in the MatLab environment and the Java programming language. Ill. 11, bibliogr. 23.

Keywords: quadratic programming, projecting a point to a simplex, optimality conditions.

REFERENCES

1. M. K. Gavurin and V. N. Malozemov, *Ekstremal'nye zadachi s lineinymi ograniicheniyami* (Extreme Problems with Linear Constraints), Izd. Leningr. Univ., Leningrad, 1984.
2. V. F. Demyanov and G. Sh. Tamasyan, On direct methods for solving variational problems, *Tr. Inst. Mat. Mekh.*, **16**, No. 5, 36–47, 2010.
3. M. V. Dolgopolik and G. Sh. Tamasyan, On equivalence of the method of steepest descent and the method of hypodifferential descent in some constrained optimization problems, *Izv. Sarat. Univ., Ser. Mat. Mekh. Inform.*, **14**, No. 4-2, 532–542, 2014.
4. V. N. Malozemov, MDM method — 40 years, *Vestn. Syktyvkar. Univ., Ser. 1*, No. 15, 51–62, 2012.
5. V. N. Malozemov and A. B. Pevnyi, Fast algorithm for projecting a point on a simplex, *Vestn. St.-Peterbg. Univ., Ser. 1*, No. 1, 112–113, 1992. Translated in *Vestn. St. Petersburg Univ., Math.*, **25**, No. 1, 62–63, 1992.
6. V. N. Malozemov and G. Sh. Tamasyan, Two fast algorithms for finding the projection of a point onto the standard simplex, *Zh. Vychisl. Mat. Mat. Fiz.*, 2016. DOI: 10.7868/S0044466916050148

7. **G. Sh. Tamasyan**, Methods of steepest and hypodifferential descent in one problem of calculus of variations, *Vychisl. Metody Program.*, **13**, No. 1, 197–217, 2012.
8. **G. Sh. Tamasyan**, Numerical methods in problems of calculus of variations for functionals depending on higher order derivatives, in *Problemy matematicheskogo analiza* (Problems of Mathematical Analysis), Vol. 67, pp. 113–132, Tamara Rozhkovskaya, Novosibirsk, 2012. Translated in *J. Math. Sci.*, **188**, No. 3, 299–321, 2013.
9. **G. Sh. Tamasyan** and **A. A. Chumakov**, Finding the distance between ellipsoids, *Diskretn. Anal. Issled. Oper.*, **21**, No. 3, 87–102, 2014. Translated in *J. Appl. Ind. Math.*, **8**, No. 3, 400–410, 2014.
10. **A. Yu. Uteshev** and **M. V. Yashina**, Computation of the distance from an ellipsoid to a linear surface and a quadric in \mathbb{R}^n , *Dokl. Akad. Nauk*, **419**, No. 4, 471–474, 2008. Translated in *Dokl. Math.*, **77**, No. 2, 269–272, 2008.
11. **P. Brucker**, An $O(n)$ algorithm for quadratic knapsack problems, *Oper. Res. Lett.*, **3**, No. 3, 163–166, 1984.
12. **A. Causa** and **F. Raciti**, A purely geometric approach to the problem of computing the projection of a point on a simplex, *J. Optimization Theory Appl.*, **156**, No. 2, 524–528, 2013.
13. **V. F. Demyanov**, **F. Giannessi**, and **G. Sh. Tamasyan**, Variational control problems with constraints via exact penalization, in F. Giannessi and A. Maugeru, eds., *Variational Analysis and Applications*, pp. 301–342, Springer, New York, 2005 (Nonconvex Optim. Its Appl., Vol. 79).
14. **V. F. Demyanov** and **G. Sh. Tamasyan**, Exact penalty functions in isoperimetric problems, *Optimization*, **60**, No. 1–2, 153–177, 2011.
15. **V. F. Demyanov** and **G. Sh. Tamasyan**, Direct methods in the parametric moving boundary variational problem, *Numer. Funct. Anal. Optimization*, **35**, No. 7–9, 932–961, 2014.
16. **M. V. Dolgopolik** and **G. Sh. Tamasyan**, Method of steepest descent for two-dimensional problems of calculus of variations, in V. F. Demyanov, P. M. Pardalos, and M. Batsyn, eds., *Constructive Nonsmooth Analysis and Related Topics*, pp. 101–113, Springer, New York, 2014 (Springer Optimization Its Appl., Vol. 87).
17. **M. Held**, **P. Wolfe**, and **H. P. Crowder**, Validation of subgradient optimization, *Math. Program.*, **6**, No. 1, 62–88, 1974.
18. **R. V. Helgason**, **J. L. Kennington**, and **H. Lall**, A polynomially bounded algorithm for a singly constrained quadratic program, *Math. Program.*, **18**, No. 1, 338–343, 1980.
19. **N. Maculan** and **G. Galdino de Paula, Jr.**, A linear-time median-finding algorithm for projecting a vector on the simplex of \mathbb{R}^n , *Oper. Res. Lett.*, **8**, No. 4, 219–222, 1989.
20. **C. Michelot**, A finite algorithm for finding the projection of a point onto the canonical simplex of \mathbb{R}^n , *J. Optim. Theory Appl.*, **50**, No. 1, 195–200, 1986.

21. **M. Patriksson**, A survey on the continuous nonlinear resource allocation problem, *Eur. J. Oper. Res.*, **185**, No. 1, 1–46, 2008.
22. **G. Tamasyan, A. Chumakov**, Finding the distance between the ellipsoid and the intersection of a linear manifold and ellipsoid, in *Proc. 2015 Int. Conf. “Stability and Control Processes” in Memory of V.I. Zubov (SCP) joined with 21st Int. Workshop on Beam Dynamics and Optimization (BDO), St. Petersburg, Russia, Oct. 5–9, 2015*, pp. 357–360, IEEE, 2015. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7342138>
23. **G. Tamasyan, E. Prosolupov**, Orthogonal projection of a point onto the standard simplex algorithms analysis, in *Proc. 2015 Int. Conf. “Stability and Control Processes” in Memory of V.I. Zubov (SCP) joined with 21st Int. Workshop on Beam Dynamics and Optimization (BDO), St. Petersburg, Russia, Oct. 5–9, 2015*, pp. 353–356, IEEE, 2015. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7342137>

Grigoriy Sh. Tamasyan,

Evgenii V. Prosolupov,

Todor A. Angelov

Received

11 September 2015

Revised

19 October 2015