УДК 519.85
MSC 90C20

Вестник СПбГУ. Математика. Механика. Астрономия. 2022. Т. 9 (67). Вып. 1

Быстрый алгоритм решения квадратичной задачи о ранце

А. В. Плоткин

Санкт-Петербургский государственный университет,
Российская Федерация, 199034, Санкт-Петербург, Университетская наб., 7–9

Для цитирования: *Плоткин А. В.* Быстрый алгоритм решения квадратичной задачи о ранце // Вестник Санкт-Петербургского университета. Математика. Механика. Астрономия. 2022. Т. 9 (67). Вып. 1. С. 76–84. https://doi.org/10.21638/spbu01.2022.108

В работе рассматривается задача квадратичного программирования со строго выпуклой сепарабельной целевой функцией, единственным линейным ограничением и двусторонними ограничениями на переменные. В англоязычной литературе эта задача называется Convex Knapsack Separable Quadratic Problem, или сокращенно — CKSQP. Нас интересует алгоритм решения задачи CKSQP с линейной сложностью. Посвященные этой теме работы содержат неточности в описании алгоритмов и неэффективные реализации. В данной работе удалось преодолеть имеющиеся трудности.

Ключевые слова: квадратичное программирование, задача о ранце, быстрые алгоритмы.

1. Постановка задачи. Рассмотрим следующую задачу квадратичного программирования:

$$\begin{aligned} &\tfrac{1}{2}\langle Hx, x\rangle - \langle c, x\rangle \to \min, \\ &\sum_{i=1}^{n} a_i x_i = b, \\ &l_i \leqslant x_i \leqslant r_i, \quad i \in 1:n. \end{aligned} \tag{1}$$

Здесь b, c_i, a_i — вещественные параметры, $l_i \in \mathbb{R} \cup \{-\infty\}$, $r_i \in \mathbb{R} \cup \{+\infty\}$, и H — диагональная матрица порядка n с положительными диагональными элементами $h_1, h_2, \ldots, h_n$. Целевая функция задачи (1) является строго выпуклой и сепарабельной. Через Ω обозначим множество планов, то есть векторов $x = (x_1, x_2, \ldots, x_n)$, удовлетворяющих ограничениям задачи (1).

Данная задача рассматривалась, например, в диссертации [1], где имеется обширная библиография. В работе [2] приводится критический анализ существующих алгоритмов решения задачи (1).

В тексте статьи мы будем предполагать, что l_i, $r_i \in \mathbb{R}$. Это позволит упростить изложение. При этом разработанное программное обеспечение будет учитывать случаи $l_i = -\infty$ и $r_i = +\infty$.

Во многих приложениях задача (1) встречается с единичной матрицей H, что соответствует проектированию точки c на множество Ω [3, 4]. Из других приложений можно выделить квадратичную задачу распределений ресурсов [5] и вычисление матрицы обновления в квазиньютоновском методе при наличии ограничений на якобиан [6].

2. Упрощение постановки. Не умаляя общности, можно считать, что в задаче (1) выполнены условия $l_i < r_i$ и $a_i \neq 0$ при всех i от 1 до n. Если $a_i = 0$, то оптимальное значение x_i находится путем решения тривиальной задачи:

$$\begin{aligned} &\tfrac{1}{2}h_i x_i^2 - c_i x_i \to \min, \\ &l_i \leqslant x_i \leqslant r_i. \end{aligned}$$

Воспользовавшись предположением о том, что коэффициенты a_i в определении множества Ω отличны от нуля, выполним замену переменных:

$$x_i = \hat{x}_i/a_i + c_i/h_i, \quad i \in 1:n. \tag{2}$$

Получим задачу следующего вида:

$$\begin{aligned} &\tfrac{1}{2}\langle \hat{H}\hat{x}, \hat{x}\rangle \to \min, \\ &\sum_{i=1}^{n} \hat{x}_i = \hat{b}, \\ &\hat{l}_i \leqslant \hat{x}_i \leqslant \hat{r}_i, \quad i \in 1:n, \end{aligned} \tag{3}$$

где

$$\hat{H} = \operatorname{diag}(h_1/a_1^2, \ldots, h_n/a_n^2),$$

$$\hat{b} = b - \sum_{i=1}^{n} \frac{a_i c_i}{h_i},$$

$$\hat{l}_i = \begin{cases} a_i(l_i - c_i/h_i), & \text{если} \quad a_i > 0, \\ a_i(r_i - c_i/h_i), & \text{если} \quad a_i < 0, \end{cases}$$

$$\hat{r}_i = \begin{cases} a_i(r_i - c_i/h_i), & \text{если} \quad a_i > 0, \\ a_i(l_i - c_i/h_i), & \text{если} \quad a_i < 0. \end{cases}$$

Задача (3) является частным случаем задачи (1). Преобразование (2) имеет линейную сложность и значительно упрощает дальнейшие вычисления. В связи с этим далее будем считать, что решаемая задача имеет следующий вид:

$$\begin{aligned} &Q(x) := \tfrac{1}{2}\langle Hx, x\rangle \to \min, \\ &\sum_{i=1}^{n} x_i = b, \\ &l_i \leqslant x_i \leqslant r_i, \quad i \in 1:n. \end{aligned} \tag{4}$$

3. Условия Куна — Таккера. Целевая функция задачи (4) непрерывна и строго выпукла, а значит критерием существования решения является непустота множества планов Ω. В данном случае условие непустоты множества планов записывается следующим образом:

$$\sum_{i=1}^{n} l_i \leqslant b \leqslant \sum_{i=1}^{n} r_i. \tag{5}$$

Будем считать, что условие (5) выполнено, а значит множество планов Ω непусто.

Заметим, что множество Ω является выпуклым. Вместе со строгой выпуклостью целевой функции $Q(x)$ это гарантирует единственность решения задачи (4). Для его нахождения воспользуемся условиями Куна — Таккера:

$$\left.\begin{gathered} h_i x_i = \lambda + u_i - v_i, \\ u_i(x_i - l_i) = 0, \quad v_i(r_i - x_i) = 0, \\ u_i \geqslant 0, \quad v_i \geqslant 0, \\ l_i \leqslant x_i \leqslant r_i, \end{gathered}\right\} \quad i \in 1:n, \tag{6}$$

$$\sum_{i=1}^{n} x_i = b. \tag{7}$$

4. Характеристика оптимального плана. Разберемся с условиями (6). Для этого нам потребуется вспомогательная лемма.

Лемма. *Пусть выполнены соотношения*

$$hw = \lambda + u - v, \tag{8}$$

$$u(w - l) = 0, \quad v(r - w) = 0, \tag{9}$$

$$u \geqslant 0, \quad v \geqslant 0, \tag{10}$$

$$l \leqslant w \leqslant r,$$

где h, l, r — вещественные константы, причем $c > 0$ и $l < r$. Тогда необходимо, чтобы

$$w = \begin{cases} l, & \text{если} \quad \lambda/h \leqslant l, \\ \lambda/h, & \text{если} \quad l < \lambda/h < r, \\ r, & \text{если} \quad \lambda/h \geqslant r. \end{cases} \tag{11}$$

Доказательство. Проведем анализ полного набора альтернатив для переменной w.

(a1) $w = l$. Согласно (9), $v = 0$. В силу (8) и (10) имеем $hw = \lambda + u \geqslant \lambda$, так что

$$\lambda \leqslant hl. \tag{12}$$

(a2) $l < w < r$. Согласно (9), $u = v = 0$, так что в силу (8) имеем $\lambda = hw$. В частности,

$$\lambda \in (hl, hr). \tag{13}$$

(a3) $w = r$. Согласно (9), $u = 0$. В силу (8) и (10) имеем $hw = \lambda - v \leqslant \lambda$, так что

$$\lambda \geqslant hr. \tag{14}$$

Отметим, что условия (12)–(14) представляют собой полный набор альтернатив для переменной $\lambda \in \mathbb{R}$.

Переходим к доказательству формулы (11).

Если $\lambda/h \leqslant l$, то $\lambda \notin (hl, hr)$ и $\lambda \leqslant hl < hr$, поэтому альтернативы (a2) и (a3) исключаются. Остается одна возможность — $w = l$.

Если $l < \lambda/h < r$, то $\lambda > hl$ и $\lambda < hr$, поэтому альтернативы (a1) и (a3) исключаются. Остается одна возможность — $l < w < r$. Согласно (a2), $w = \lambda/h$.

Наконец, если $\lambda/h \geqslant r$, то $\lambda \notin (hl, hr)$ и $\lambda \geqslant hr > hl$, поэтому альтернативы (a1) и (a2) исключаются. Остается одна возможность — $w = r$.

Лемма доказана. □

Формула (11) определяет переменную w как функцию от λ: $w = w(\lambda)$. На рис. 1 изображен график функции $w(\lambda)$.

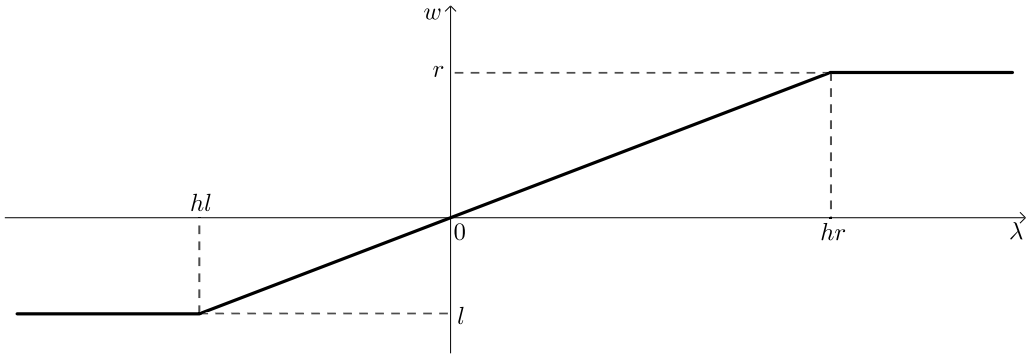

Рис. 1. График функции $w(\lambda)$.

Функция $w(\lambda)$ допускает аналитическое представление (см., например, [7])

$$w(\lambda) = l + \tfrac{1}{h}(\lambda - hl)_+ - \tfrac{1}{h}(\lambda - hr)_+, \tag{15}$$

где $(u)_+ = \max\{0, u\}$.

5. Сведение к скалярному уравнению. Вернемся к условиям Куна—Таккера (6), (7).

Воспользуемся доказанной леммой при каждом i от 1 до n. Положив $w = x_i$, $h = h_i$, $l = l_i$, $r = r_i$, $u = u_i$, $v = v_i$, получим представление x_i как функций от λ:

$$x_i(\lambda) = l_i + \tfrac{1}{h_i}(\lambda - h_i l_i)_+ - \tfrac{1}{h_i}(\lambda - h_i r_i)_+, \quad i \in 1:n.$$

Каждая из функций $x_i(\lambda)$ является непрерывной, неубывающей, имеет минимальное l_i и максимальное r_i значения.

Таким образом, условию (6) удовлетворяет множество

$$X = \left\{ \big(x_1(\lambda), \ldots, x_n(\lambda)\big) \mid \lambda \in \mathbb{R} \right\}.$$

Добавив необходимость выполнения условия (7) на этом множестве, получим следующее уравнение:

$$\sum_{i=1}^{n} x_i(\lambda) = b, \quad \lambda \in \mathbb{R}. \tag{16}$$

Свойства функций $x_i(\lambda)$ вместе с условием (5) гарантируют существование решения данного уравнения.

Подведем итоги.

Теорема. *Пусть λ^* — решение уравнения* (16)*, тогда $x^* = (x_1^*, \ldots, x_n^*)$, где $x_1^* = x_1(\lambda^*), \ldots, x_n^* = x_n(\lambda^*)$, является единственным решением задачи* (4)*.*

Далее нас будет интересовать быстрый алгоритм решения уравнения (16).

6. Подготовка к построению алгоритма. Введем следующие обозначения:

$$\left.\begin{aligned} &\alpha_i = \tfrac{1}{h_i}, \\ &\alpha_{i+n} = -\tfrac{1}{h_i}, \\ &\lambda_i = h_i l_i, \\ &\lambda_{i+n} = h_i r_i, \end{aligned}\right\} \quad i \in 1:n,$$

$$\phi_i(\lambda) = \alpha_i(\lambda - \lambda_i)_+, \quad i \in 1:2n,$$

$$F(\lambda) = \sum_{i=1}^{2n} \phi_i(\lambda),$$

$$C = b - \sum_{i=1}^{n} l_i.$$

Тогда уравнение (16) можно переписать в следующем виде:

$$F(\lambda) = C. \tag{17}$$

При этом функция $F(\lambda)$ является неубывающей ломаной с узлами $\lambda_1, \ldots, \lambda_{2n}$.

Во избежание двусмысленности дадим определение медиане, которого будем придерживаться в дальнейшем.

Определение. Пусть задано конечное множество вещественных чисел $A = \{a_i\}_{i=1}^m$. Обозначим через $b_1, b_2, \ldots, b_m$ последовательность, полученную в результате упорядочивания элементов множества A в порядке неубывания. Тогда элемент $b_{\lceil \frac{m+1}{2} \rceil}$ называется *медианой* множества A.

Напомним, что медиану множества A можно найти за линейное время [8, с. 250–253].

Перейдем к построению алгоритма.

Условие существования решения гарантирует нам выполнение неравенства

$$F(\min_{i \in 1:2n} \lambda_i) \leqslant C \leqslant F(\max_{i \in 1:2n} \lambda_i).$$

Наш алгоритм будет вычислять значения функции F исключительно в узлах λ_i, а в качестве результата будет возвращать узел $\hat{\lambda} = \max\{\lambda_i \mid F(\lambda_i) \leqslant C\}$. Имея такой узел, несложно найти ближайший к нему узел справа (если такой имеется), вычислить значение функции в обоих узлах и выполнить обратную интерполяцию для нахождения λ^*. Такие дополнительные действия требуют линейного времени работы и не окажут влияния на итоговую вычислительную сложность алгоритма.

Для упрощения понимания сначала мы представим описание алгоритма, работающего за время $O(n \log n)$, а затем покажем, как добиться линейного времени работы.

7. Упрощенная версия — алгоритм 1. Опишем упрощенную версию алгоритма.

Начальный шаг. Положим $I_0 = \{1, 2, \ldots, 2n\}$.

Инвариант. Искомый узел $\hat{\lambda}$ содержится во множестве Λ_k, где $\Lambda_k = \{\lambda_i\}_{i \in I_k}$.

Шаг алгоритма. Пусть имеется множество I_k. Если множество I_k содержит единственный элемент, то этот элемент является индексом искомого узла $\hat{\lambda}$. Иначе, найдем медиану множества Λ_k и обозначим этот узел через λ_{med}, где med — индекс в I_k. Вычислим значение функции: $F_{\text{med}} = F(\lambda_{\text{med}})$. Если $F_{\text{med}} \leqslant C$, то значение $\hat{\lambda}$ точно не меньше λ_{med}. Если же $F_{\text{med}} > C$, то $\hat{\lambda}$ точно меньше λ_{med}.

Таким образом, положим

$$I_{k+1} = \begin{cases} \{\text{med}\} \cup \{i \in I_k \mid \lambda_i > \lambda_{\text{med}}\}, & \text{если } F_{\text{med}} \leqslant C, \\ \{i \in I_k \mid \lambda_i < \lambda_{\text{med}}\}, & \text{если } F_{\text{med}} > C. \end{cases}$$

Полученное множество I_{k+1} является подмножеством I_k и имеет размер, не превышающий $\left\lceil \frac{|I_k|}{2} \right\rceil$.

Оценка сложности. Алгоритм делает $O(\log n)$ итераций. На каждой итерации алгоритма выполняется вычисление функции F за $O(n)$ операций. Затраты на вычисление медианы суммарно составляют $O(2n) + O(n) + O(\frac{n}{2}) + O(\frac{n}{4}) + \ldots + O(1) = O(n)$. Таким образом, итоговая сложность алгоритма равна $O(n \log n)$.

8. Быстрый алгоритм — алгоритм 2. Для достижения линейной сложности необходимо научиться вычислять значение целевой функции как можно быстрее. Добъемся того, чтобы вычисление функции F на k-й итерации требовало $O(|I_k|)$ операций.

Сделаем следующее наблюдение. Возьмем произвольный индекс i_0. Пусть в какой-то момент этот индекс вышел из последовательности множеств I: $i_0 \in I_k$, $i_0 \notin I_{k+1}$. Тогда $\lambda_{i_0} \leqslant \min \Lambda_{k+1}$ или $\lambda_{i_0} \geqslant \max \Lambda_{k+1}$ по построению. Так как все дальнейшие вычисления функции F будут происходить только в узлах I_{k+1}, то уже на этапе исключения индекса i_0 можно однозначно описать вклад функции $\phi_{i_0}(\lambda)$ в $F(\lambda)$. Если $\lambda_{i_0} \leqslant \min \Lambda_{k+1}$, то $\phi_{i_0}(\lambda) = \alpha_{i_0}\lambda - \alpha_{i_0}\lambda_{i_0}$ для всех $\lambda \in I_{k+1}$. Если $\lambda_{i_0} \geqslant \max \Lambda_{k+1}$, то $\phi_{i_0}(\lambda) = 0$ для всех $\lambda \in I_{k+1}$.

Введем и будем поддерживать линейную функцию $A_k\lambda + B_k$, в которой будут «аккумулироваться» линейные функции узлов, вышедших из рассмотрения.

Начальный шаг. Положим $I_0 = \{1, 2, \ldots, 2n\}$, $A_0 = 0$, $B_0 = 0$.

Инвариант 1. Искомый узел $\hat{\lambda}$ содержится во множестве Λ_k, где $\Lambda_k = \{\lambda_i\}_{i \in I_k}$.

Инвариант 2. Для всех узлов λ из I_k значение функции F может быть вычислено по формуле $F(\lambda) = \sum_{i \in I_k} \phi_i(\lambda) + A_k\lambda + B_k$.

Шаг алгоритма. Пусть имеется множество I_k. Если множество I_k содержит единственный элемент, то этот элемент является индексом искомого узла $\hat{\lambda}$. Иначе, найдем медиану множества Λ_k и обозначим этот узел через λ_{med}, где med — индекс в I_k. Вычислим значение функции: $F_{\text{med}} = \sum_{i \in I_k} \phi_i(\lambda_{\text{med}}) + A_k\lambda_{\text{med}} + B_k$. Если $F_{\text{med}} \leqslant C$, то значение $\hat{\lambda}$ не меньше λ_{med}. Если же $F_{\text{med}} > C$, то $\hat{\lambda}$ точно меньше λ_{med}.

В случае $F_{\text{med}} \leqslant C$ положим

$$\begin{aligned} I_{k+1} &= \{\text{med}\} \cup \{i \in I_k \mid \lambda_i > \lambda_{\text{med}}\}, \\ A_{k+1} &= A_k + \sum_{i \in I_k \setminus I_{k+1}} \alpha_i, \\ B_{k+1} &= B_k - \sum_{i \in I_k \setminus I_{k+1}} \alpha_i\lambda_i. \end{aligned}$$

В случае $F_{\text{med}} > C$ положим

$$I_{k+1} = \{i \in I_k \mid \lambda_i < \lambda_{\text{med}}\},$$
$$A_{k+1} = A_k,$$
$$B_{k+1} = B_k.$$

Оценка сложности. Теперь затраты на вычисление функций, пересчет коэффициентов и работу со множествами составляют $O(2n) + O(n) + O(\frac{n}{2}) + O(\frac{n}{4}) + \ldots + O(1) = O(n)$. Таким образом, итоговая сложность алгоритма равна $O(n)$.

9. Численные эксперименты. Проведем серию численных экспериментов.

Будем решать задачи вида (4) для разных n: $10^4, 10^5, 10^6, 10^7$. Коэффициенты h будем выбирать случайно из отрезка $[0.1, 1]$, коэффициенты l и r — из отрезка $[-1, 1]$. Параметр b выберем случайно из отрезка $\left[\sum_{i=1}^{n} l_i, \sum_{i=1}^{n} r_i\right]$.

В нашем алгоритме для поиска медианы мы будем использовать алгоритм Quickselect [8, с. 245–249], осуществляющий поиск за $O(n)$ в среднем, но показывающий хорошие результаты на практике.

Для каждого из рассматриваемых n были сгенерированы 100 тестовых задач. Вычисления производились на машине с процессором AMD Ryzen 5 3500U и 16 Гб ОЗУ. В табл. 1 приводится среднее время работы обеих версий алгоритма.

Таблица 1. **Среднее время работы алгоритмов (мс) для решения задачи вида (4)**

n	Алгоритм 1: $O(n \log n)$	Алгоритм 2: $O(n)$
10^4	2	1
10^5	16	11
10^6	105	71
10^7	1175	720

Далее было решено провести сравнение скорости работы полученной реализации со специализированной библиотекой алгоритмов CQKnPClass Project [9]. Один из представленных в библиотеке алгоритмов под названием DualCQKnP ориентирован на решение задач вида (1).

Сравнительный анализ производился с помощью тестовой процедуры MainRnd, использовавшейся авторами статьи [9].

Для n, равного 10^4, 10^5, 10^6, 10^7, были созданы 100 тестовых задач. Некоторые из них могли иметь пустое множество планов. В табл. 2 приводится среднее время работы сравниваемых алгоритмов.

Таблица 2. **Среднее время работы алгоритмов (мс) для решения задачи вида (1)**

n	`DualCQKnP`	Алгоритм 1: $O(n \log n)$	Алгоритм 2: $O(n)$
10^4	1	1	1
10^5	20	11	4
10^6	303	144	43
10^7	4552	1588	486

Стоит отметить, что алгоритм DualCQKnP имеет вычислительную сложность $O(n \log n)$, и авторы ставили под сомнение возможность эффективной реализации алгоритма с линейным временем работы.

Реализация полученных алгоритмов и тестовые процедуры доступны в репозитории [10].

Литература

1. Jeong J. *Indefinite Knapsack Separable Quadratic Programming: Methods and Applications*. Dr. Sci. thesis (2014).

2. Kiwiel K. Breakpoint searching algorithms for the continuous quadratic knapsack problem. *Mathematical Programming* **112**, 473–491 (2008). https://doi.org/10.1007/s10107-006-0050-z

3. Dai Y. H., Fletcher R. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Mathematical Programming* **106**, 403–421 (2006). https://doi.org/10.1007/s10107-005-0595-2

4. Nielsen S., Zenios S. Massively parallel algorithms for singly-constrained convex programs. *ORSA J. Comput.* **4**, 166–181 (1992). https://doi.org/10.1287/ijoc.4.2.166

5. Patriksson M. A survey on the continuous nonlinear resource allocation problem. *European J. Operational Research* **185**, 1–46 (2008). https://doi.org/10.1016/j.ejor.2006.12.006

6. Calamai P., Moré J. Quasi-Newton updates with bounds. *SIAM J. Numer. Anal.* **24**, 1434–1441 (1987).

7. Малозёмов В. Н., Тамасян Г. Ш. Представления непрерывных кусочно-аффинных функций. В: *Семинар «CNSA & NDO». Избранные доклады. 10 сентября 2019 г.* (2019).

8. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. *Алгоритмы. Построение и анализ*, пер. с англ. 3-е изд. Вильямс (2013).

9. Frangioni A., Gorgone E. A library for continuous convex separable quadratic knapsack problems. *European J. Operational Research* **229**, 37–40 (2013). https://doi.org/10.1016/j.ejor.2013.02.038

10. GitHub. Доступно на: https://github.com/WhatElseIsThere/CKSQP (дата обращения: 30.11.2021).

Статья поступила в редакцию 10 июня 2021 г.;
доработана 16 июня 2021 г.;
рекомендована к печати 2 сентября 2021 г.

Контактная информация:

Плоткин Артем Владимирович — аспирант; avplotkin@gmail.com

Fast algorithm for quadratic knapsack problem

A. V. Plotkin

St Petersburg State University, 7–9, Universitetskaya nab., St Petersburg, 199034, Russian Federation

For citation: Plotkin A. V. Fast algorithm for quadratic knapsack problem. *Vestnik of Saint Petersburg University. Mathematics. Mechanics. Astronomy*, 2022, vol. 9 (67), issue 1, pp. 76–84. https://doi.org/10.21638/spbu01.2022.108 (In Russian)

The paper considers a quadratic programming problem with a strictly convex separable objective function, a single linear constraint, and two-sided constraints on variables. This problem is commonly called the Convex Knapsack Separable Quadratic Problem, or CKSQP for short. We are interested in an algorithm for solving CKSQP with a linear time complexity. The papers devoted to this topic contain inaccuracies in the description of algorithms and ineffective implementations. In this work, the existing difficulties were overcome.

Keywords: quadratic programming, knapsack problem, fast algorithms.

References

1. Jeong J. *Indefinite Knapsack Separable Quadratic Programming: Methods and Applications.* Dr. Sci. thesis (2014).

2. Kiwiel K. Breakpoint searching algorithms for the continuous quadratic knapsack problem. *Mathematical Programming* **112**, 473–491 (2008). https://doi.org/10.1007/s10107-006-0050-z

3. Dai Y. H., Fletcher R. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Mathematical Programming* **106**, 403–421 (2006). https://doi.org/10.1007/s10107-005-0595-2

4. Nielsen S., Zenios S. Massively parallel algorithms for singly-constrained convex programs. *ORSA J. Comput.* **4**, 166–181 (1992). https://doi.org/10.1287/ijoc.4.2.166

5. Patriksson M. A survey on the continuous nonlinear resource allocation problem. *European J. Operational Research* **185**, 1–46 (2008). https://doi.org/10.1016/j.ejor.2006.12.006

6. Calamai P., Moré J. Quasi-Newton updates with bounds. *SIAM J. Numer. Anal.* **24**, 1434–1441 (1987).

7. Malozemov V. N., Tamasyan G. S. Representations of continuous piecewise affine functions. In: *Seminar "CNSA & NDO". Selected reports. Sept. 10, 2019* (2019). (In Russian)

8. Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. *Introduction to Algorithms.* 3rd ed. The MIT Press (2009). [Rus. ed.: Cormen T. H., Leiserson C. E., Rivest R. L., Stein C. *Algoritmy. Postroenie i analiz.* Williams Publ. (2013)].

9. Frangioni A., Gorgone E. A library for continuous convex separable quadratic knapsack problems. *European J. Operational Research* **229**, 37–40 (2013). https://doi.org/10.1016/j.ejor.2013.02.038

10. GitHub. Available at: https://github.com/WhatElseIsThere/CKSQP (accessed: November 30, 2021).

Received: June 10, 2021
Revised: June 16, 2021
Accepted: September 2, 2021

Author's information:

Artyom V. Plotkin — avplotkin@gmail.com