УДК 004.021 doi: 10.15827/0236-235X.142.539-550 2023. Т. 36. № 4. С. 539–550

Поиск решения задачи целочисленного программирования с помощью итеративного округления координат

А.В. Иванов
Ю.Н. Матвеев

Ссылка для цитирования
Иванов А.В., Матвеев Ю.Н. Поиск решения задачи целочисленного программирования с помощью итеративного округления координат // Программные продукты и системы. 2023. Т. 36. № 4. С. 539–550. doi: 10.15827/0236-235X.142.539-550
Информация о статье
Поступила в редакцию: 24.04.2023 После доработки: 02.05.2023 Принята к публикации: 17.05.2023

Аннотация. В статье предлагается алгоритм поиска целочисленного решения, использующий идею округления координат точки оптимального нецелочисленного решения и построения луча, направленного вглубь области допустимого решения. Алгоритм основан на итеративном процессе округления координат точки в направлении построенного луча. В ходе исследования обнаружено, что движение в сторону направления луча без перебора всех возможных вариантов упрощает алгоритм и позволяет избежать ветвления. Это выделяет данный подход из других существующих на данный момент открытых методов, таких как методы отсечений и ветвей и границ. В процессе работы осуществлялись описание и экспериментальная проверка данного алгоритма и возможности его применения при разных конфигурациях области допустимых решений. Теоретическая значимость исследования заключается в разработке нового алгоритма, который не требует выполнения симплекс-метода на каждом этапе и на каждом шаге использует луч вместо плоскости, что предотвращает рост пространственной сложности задачи по сравнению с другими методами. В ходе исследования стало видно, что предложенный алгоритм имеет ограничения, однако основная идея доказала свою работоспособность, и в дальнейшем планируется развивать ее.
Ключевые слова: математическое программирование, линейное программирование, целочисленное программирование, оптимизация, алгоритм, покоординатный спуск

Введение. Задачи *целочисленного программирования* (ЦП) широко применяются в различных отраслях, таких как производство, логистика, экономика, информационные технологии и многих других [1–3]. Их актуальность обусловлена необходимостью оптимизации различных процессов и распределения ресурсов. Являясь частным случаем *линейного программирования* (ЛП) [4], ЦП имеет достаточно долгую историю [5, 6].

В данной работе развивается идея из [7], где предложен алгоритм для поиска целочисленного решения по аналогии с методом покоординатного спуска [8]. Представленный алгоритм, помимо итеративного округления координат на каждом шаге, проверяет выход за *область допустимых решений* (ОДР) и при необходимости возвращает текущее решение на ее границу.

Современное состояние предметной области

Одной из основных проблем при решении задач ЦП до сих пор является их высокая вычислительная сложность. Это обусловлено в основном тем, что точные методы решения данного рода задач, такие как метод отсечений и метод ветвей и границ [9] (также известный как метод Гомори [10]), тратят много вычислительных ресурсов впустую из-за своей природы: решают множество конфигураций одной и той же задачи (тем самым разветвляясь), что экспоненциально увеличивает время выполнения. Кроме того, они полагаются на достаточно затратную операцию нахождения оптимального решения с помощью симплекс-метода [11] на каждом шаге выполнения алгоритма, что приводит к значительным временным затратам.

Для исследования дискретных задач ЦП с логическими переменными имеется метод L-разбиений [12], который позволяет решать комбинаторные задачи, но не получать численные решения. С другой стороны, существуют эвристические алгоритмы, такие как поиск с восхождением к вершине [13] и алгоритм имитации отжига [14], являющийся примером метода Монте-Карло, а также другие подходы с применением алгоритмов машинного обучения [15]. Данные методы часто используются для решения задач ЦП из-за их более высокой скорости выполнения по сравнению с точными алгоритмами. Поскольку данные алгоритмы эвристические, в них изначально заложен элемент случайности, поэтому, используя их, приходится делать некоторые допущения в плане точности.

В одной из недавних работ [16] представлен подход к решению задач математического программирования с помощью R-функционального моделирования на основе построения вок-

сельных моделей геометрических объектов. Данный подход также увеличивает размерность пространства и в большей степени нацелен на задачи нелинейного программирования. В настоящий момент нет точных сведений о его работе с задачами ЦП.

Таким образом, можно сделать вывод о том, что существующие на данный момент методы решения задач ЦП имеют свои недостатки, обусловленные прежде всего выполнением достаточно большого количества излишних операций из-за увеличения размерности пространства решения или ограничения применения. Отличительной особенностью предлагаемого алгоритма является то, что он позволяет выполнить задачу без увеличения размерности пространства решения за счет избегания ввода новых ограничений, что потенциально может ускорить поиск.

Алгоритм поиска решения с помощью итеративного округления координат

В данной статье описывается точный алгоритм для поиска решения задачи ЦП, рассмотрены и экспериментально проверены несколько подходов. Основная идея каждого из них – смещение вглубь ОДР. Это достигается за счет построения луча, смотрящего внутрь ОДР в сторону уменьшения значения целевой функции (при задаче максимизации) и определяющего направление движения при смещении вглубь ОДР. Проверим некоторые предположения, а именно:

– определив новый базис в точке оптимального решения O, можно построить луч, который смотрел бы внутрь ОДР; данный луч можно использовать при смещении вглубь ОДР во время поиска целочисленного решения;

– поочередно округляя каждую из координат точки нецелочисленного оптимального решения O до ближайшего целого числа, можно найти целочисленное решение внутри ОДР.

Во всех случаях строится луч, выходящий из оптимальной точки O, считающейся начальной в данном алгоритме. После этого на построенном луче находится точка P, одна из координат которой является целочисленной. Из этой точки начинается итеративный процесс поиска целочисленного решения. Алгоритм состоит из нескольких шагов.

Определение ближайшего базиса

Для построения луча необходимо определить ближайшие смежные ограничения на точку O. Для этого необходимо решить уравнение ограничений вида

$$A\bar{x} - \bar{b} = 0, \quad (1)$$

определенное матрицей A и вектором $\bar{b}$ в соответствии с уравнением, где координатами $\bar{x}$ являются координаты точки O. Выполняющиеся уравнения и будут определять ограничения, которые образуют новый базис.

Допустим, даны следующие ограничения:

$$\begin{cases} -x_1 + 6x_2 \le 15, \\ 5x_1 - x_2 \le 11, \\ x_1 \ge 0, \\ x_2 \ge 0. \end{cases}$$

Тогда можно определить матрицу A и вектор $\bar{b}$:

$$A = \begin{pmatrix} 1 & -6 \\ -5 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \bar{b} = \begin{pmatrix} -15 \\ -11 \\ 0 \\ 0 \end{pmatrix}.$$

После этого, подставив точку оптимального нецелочисленного решения $O = \left(\frac{81}{29}, \frac{86}{29}\right)$ в уравнение (1), получим

$$AO - \bar{b} = \begin{pmatrix} 1 & -6 \\ -5 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 81/29 \\ 86/29 \end{pmatrix} - \begin{pmatrix} -15 \\ -11 \\ 0 \\ 0 \end{pmatrix} =$$

$$= \frac{81}{29}\begin{pmatrix} 1 \\ -5 \\ 1 \\ 0 \end{pmatrix} + \frac{86}{29}\begin{pmatrix} -6 \\ 1 \\ 0 \\ 1 \end{pmatrix} - \begin{pmatrix} -15 \\ -11 \\ 0 \\ 0 \end{pmatrix} =$$

$$= \begin{pmatrix} -15 \\ -11 \\ 81/29 \\ 86/29 \end{pmatrix} - \begin{pmatrix} -15 \\ -11 \\ 0 \\ 0 \end{pmatrix} =$$

$$= \begin{pmatrix} 0 \\ 0 \\ 81/29 \\ 86/29 \end{pmatrix}.$$

Видно, что элементы с номерами $k = 1, 2$ являются наименьшими (равны 0), а значит, соответствующие ограничения образуют новый базис в точке O (рис. 1):

$$\begin{cases} -x_1 + 6x_2 \le 15 \\ 5x_1 - x_2 \le 11 \end{cases} = \begin{cases} s_1 = 0, \\ s_2 = 0. \end{cases}$$

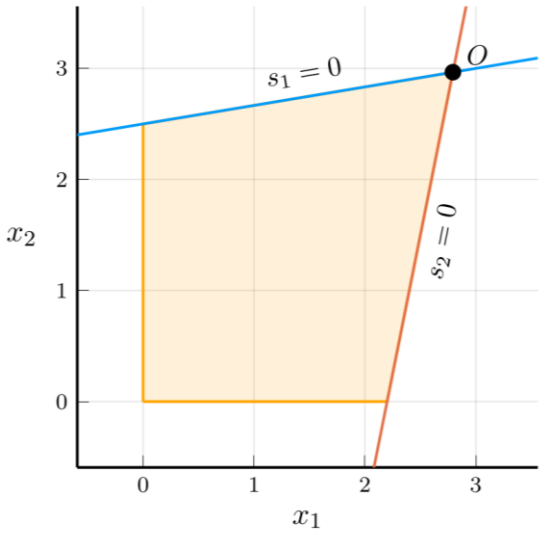


Рис. 1. Смежные ограничения, ближайшие к точке O в исходном базисе x_1x_2

Fig. 1. Intersecting constraints closest to the point O in primal basis x_1x_2

Новый базис s_1s_2 будет представлять собой прямоугольную систему координат с осями s_1 и s_2 соответственно.

Определим прямую функцию перехода от исходного базиса к новому:

$$L(\bar{x}) = M\bar{x} - \overline{b^k} \qquad (2)$$

и соответствующую ей обратную функцию перехода из нового базиса к исходному:

$$L^{-1}(\bar{x}) = M^{-1}(\bar{x} + \overline{b^k}),$$

где $\bar{x}$ – преобразуемая точка в исходном базисе; $\overline{b^k}$ – вектор k-х свободных членов; M – квадратная матрица перехода к новому базису со строками k из матрицы A:

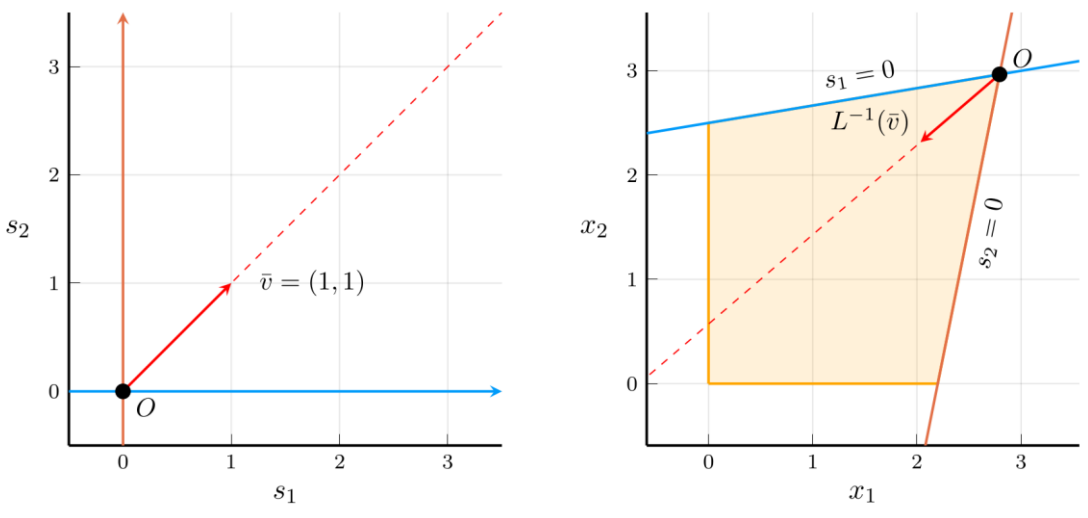
$$A = \begin{pmatrix} 1 & -6 \\ -5 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{matrix} \\ \\ \leftarrow k=1 \\ \leftarrow k=2 \end{matrix} \rightarrow M = \begin{pmatrix} 1 & -6 \\ -5 & 1 \end{pmatrix},$$

$$\bar{b} = \begin{pmatrix} -15 \\ -11 \\ 0 \\ 0 \end{pmatrix} \begin{matrix} \leftarrow k=1 \\ \leftarrow k=2 \\ \\ \end{matrix} \rightarrow \overline{b^k} = \begin{pmatrix} -15 \\ -11 \end{pmatrix}.$$

Построение луча

Наивный метод. В новом базисе M построим вектор $\bar{v} = (1,1,...,n)$ длиной n, где n – число столбцов матрицы M (равное числу измерений). Данный вектор будет определять луч в новом базисе и делить область между ограничениями пополам (рис. 2а). Зная матрицу M и вектор $\bar{v}$, можно перенести данный луч в исходный базис (рис. 2б), используя уравнение (2). Назовем такой метод построения луча наивным, поскольку простой перенос луча в исходный базис не позволяет получить луч, который делил бы ОДР ровно пополам.

Разделение области между ограничениями ровно пополам. Поскольку наивный метод построения луча не позволяет получить такой



а) *б)*

Рис. 2. Луч в новом (а) и исходном (б) базисах

Fig. 2. Ray in new basis (a) and primal basis (б)

луч, который делил бы область между базисными ограничениями ровно пополам, необходимо модифицировать метод, с помощью которого строится луч. Это становится особенно заметным, если изменить ОДР, сделав ее более узкой.

Дело в том, что масштаб координатной сетки в новом базисе не равен масштабу в старом. Из-за этого, например, вектор (1,1) в исходном базисе будет перекошен в новом базисе и наоборот. Для избежания этого нужно построить вектор $\bar{v}$ таким образом, чтобы его координатами был шаг сетки по каждой из базисных осей (в данном случае – s_1 и s_2).

Шаг сетки базисной оси – это длина нормального вектора ограничения, образующего базис. Допустим, даны ограничения:

$$\begin{cases} 8x_1 - 6x_2 \le 1 \\ -4x_1 + 2x_2 \le 3 \end{cases} = \begin{cases} s_1 = 0, \\ s_2 = 0. \end{cases}$$

Тогда их нормальные векторы равны:

$$\overline{n_{s_1}} = (8, -6),$$

$$\overline{n_{s_2}} = (-4, 2),$$

длины этих векторов соответственно:

$$\left\|\overline{n_{s_1}}\right\| = 10,$$

$$\left\|\overline{n_{s_2}}\right\| = 2\sqrt{5},$$

а вектор, задающий луч в новом базисе, можно определить следующим образом:

$$\hat{v} = \begin{pmatrix} \left\|\overline{n_{s_1}}\right\| \\ \left\|\overline{n_{s_2}}\right\| \end{pmatrix}.$$

Для удобства представления вектор $\hat{v}$ можно нормализовать:

$$\hat{v} = \frac{\bar{v}}{\left|\bar{v}\right|}.$$

Определение точки *P* на луче

Для нахождения точки, из которой будет начинаться поиск, необходимо округлить координаты точки $O = \left(\frac{81}{29}, \frac{86}{29}\right)$ вниз (в задаче максимизации): $O' = \lfloor O \rfloor = (2, 2)$.

После этого через полученную точку нужно провести секущие плоскости $x_1 = 2$, $x_2 = 2$, параллельные осям координат, и найти точки пересечения P_i между плоскостями и лучом (рис. 3). Таким образом, получается набор потенциальных начальных точек внутри ОДР, в каждой из которых одна из координат является целочисленной. Затем необходимо выбрать наилучшую точку, рассчитав значения целевой функции $f(x)$ для каждой точки P_i.

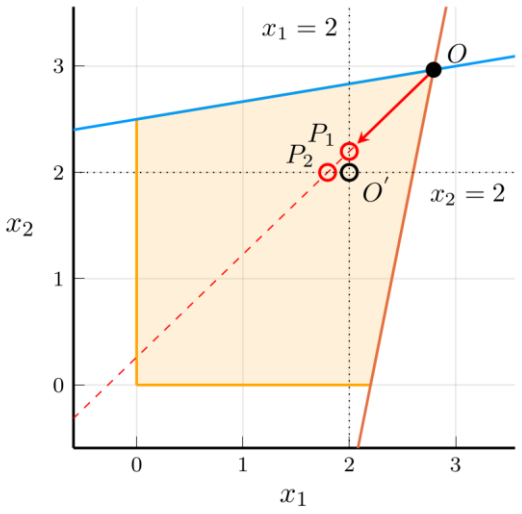


Рис. 3. Исправленный луч, секущие плоскости в точке O′ и потенциальные начальные точки P_i на луче

Fig. 3. Fixed ray, intersecting planes at point O' and potential starting points P_i on the ray

Допустим, задана целевая функция $f(\bar{x}) = 3x_1 + 2x_2$, тогда можно найти ее значения:

$$f(P_1) = 3 \cdot 2 + 2 \cdot \frac{16}{7} = \frac{74}{7} \approx 10.57,$$

$$f(P_2) = 3 \cdot \frac{5}{3} + 2 \cdot 2 = 9.$$

Значение целевой функции для P_1 наибольшее, значит, точка $P = P_1$ будет начальной.

Выбор оси координат для округления

После нахождения новой начальной точки $P = \left(2, \frac{16}{17}\right)$ необходимо определить ось, вдоль которой будет происходить смещение. Поскольку у точки P одна из координат уже является целочисленной, выбирается ось координат, по которой из других осей следует двигаться далее. Так как в данном случае всего две оси (x_1 и x_2) и известно, что координата x_1 точки P уже целочисленная, выбора не остается и смещение будет происходить вдоль оси x_2.

Смещение точки *P* вдоль выбранной оси координат

На первом этапе координата x_1 точки P целочисленная, следовательно, нужно округлять

по следующей доступной координате – x_2. Обозначим точку с округленной координатой как $\lceil P \rceil$. Перенесем точку $\lceil P \rceil$ в новый базис и обозначим как $L(\lceil P \rceil)$. Одна из ее координат (s_2) отрицательная. Это означает, что точка вышла за границы ОДР. Чтобы найти точку на границе ОДР, можно провести прямую между точками P и $L(\lceil P \rceil)$ и найти ее пересечение с осью s_1. Полученную точку обозначим $\lceil P \rceil'$ (рис. 4) и перенесем обратно в исходный базис, обозначив как $L^{-1}(\lceil P \rceil')$ – это будет следующей точкой для поиска решения.

Аналогично округлим начальную точку P вниз. Все ее координаты становятся целочисленными и положительными в новом базисе (рис. 5), следовательно, целочисленное решение для данного примера найдено.

После выполнения данных шагов координаты точки P проверяются на целочисленность. Если хотя бы одна из координат не является целочисленной, все шаги повторяются для новой точки P. Целочисленное решение будет найдено, когда все координаты точки P станут целочисленными. На рисунке 6 представлена общая блок-схема описанного алгоритма.

Непосредственно сама процедура одного смещения точки получает на вход точку P, индекс оси i, вдоль которой происходит смещение, а также направление округления – вверх или вниз. Процедура должна сделать копию точки P в памяти, которую обозначим Q, поскольку в дальнейшем понадобятся измененная точка Q и изначальная точка P. На первом шаге проверяется, является ли i-я координата точки Q целым числом. В случае целого числа

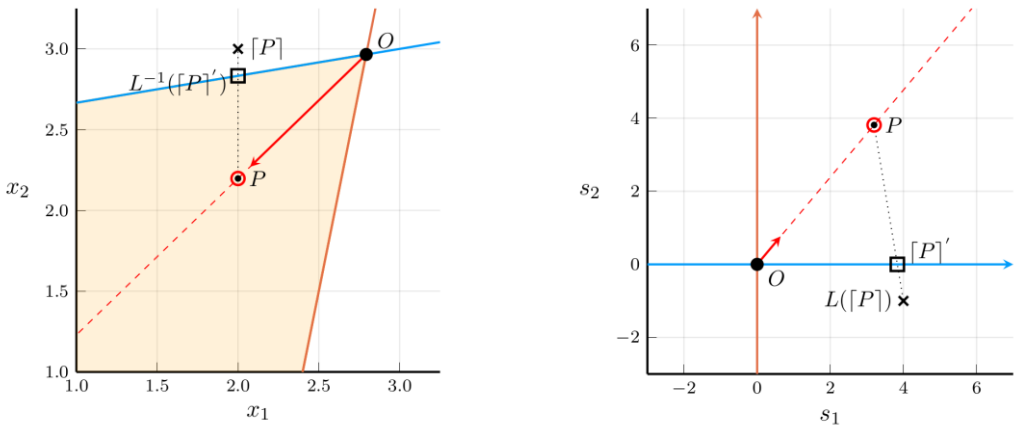


Рис. 4. Округление координаты x_2 точки P вверх и определение ее выхода за ОДР

Fig. 4. Rounding up coordinate x_2 of point P and checking if point P left the feasible region

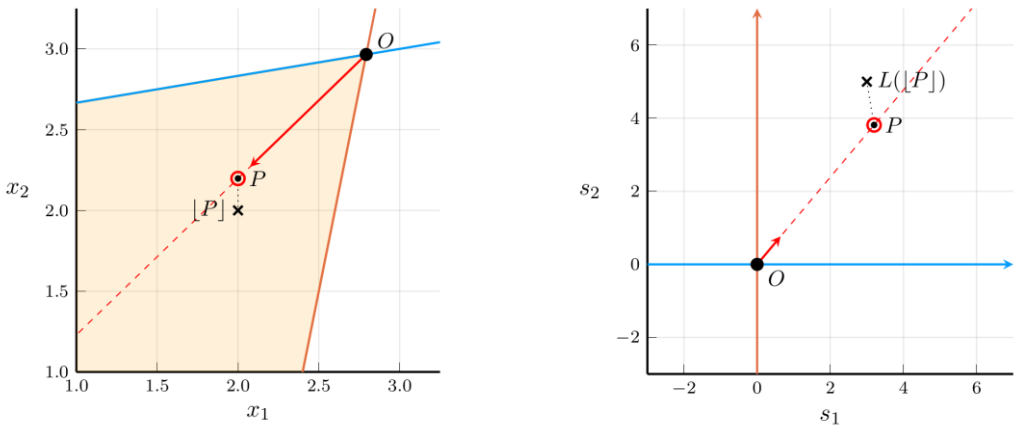


Рис. 5. Округление координаты x_2 точки P вниз и определение ее выхода за ОДР

Fig. 5. Rounding down coordinate x_2 of point P and checking if point P left the feasible region

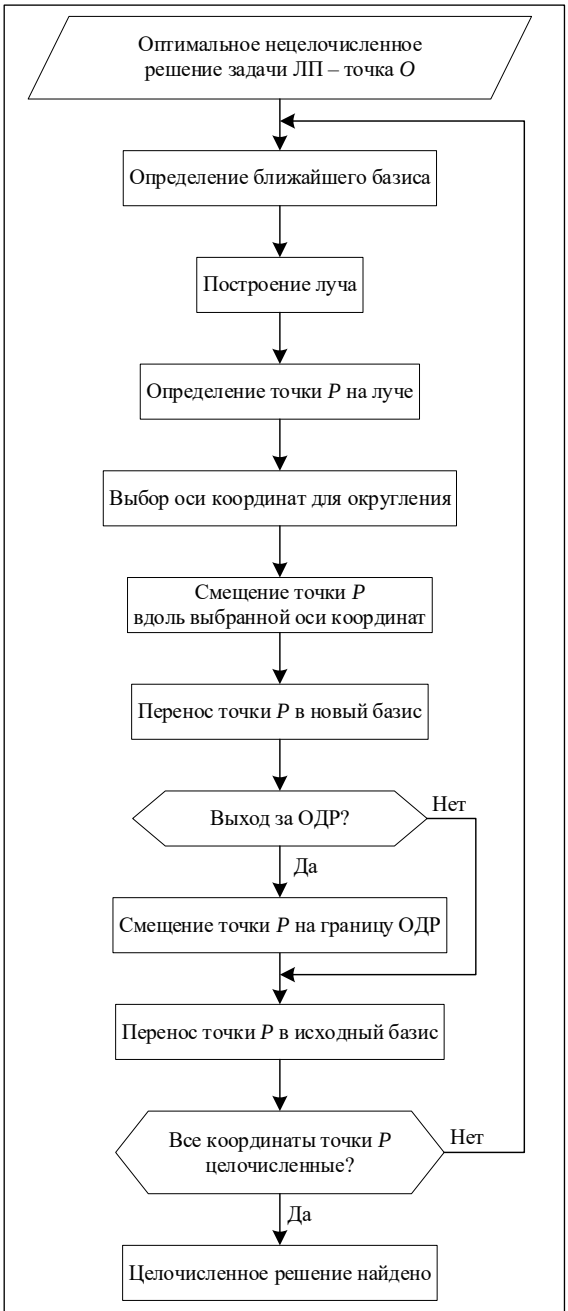


Рис. 6. Общая блок-схема алгоритма

Fig. 6. Flow chart of the algorithm

производится ее смещение на единицу в положительную сторону вдоль оси i в случае округления вверх или же в отрицательную в случае округления вниз. Если i-я координата точки Q не является целым числом, ее значение округляется либо вверх, либо вниз до ближайшего целого. После этого точки P и Q переносятся в новый базис, найденный в начале итерации (на первом шаге алгоритма). Обозначим новые точки как $P_T = L(P)$ и $Q_T = L(Q)$.

Затем проверяются значения координат точки Q_T на неотрицательность. Отрицательное значение указывает на выход за ОДР. В таком случае осуществляется поиск точки пересечения между прямой P_TQ_T и i-й осью координат в новом базисе. Обозначим данную точку как R и перенесем ее в исходный базис: $Q = L^{-1}(R)$. В результате получим точку Q, которая будет либо смещена на единицу вдоль оси i, либо находиться на границе ОДР на данной оси.

Представим процедуру выполнения одного шага алгоритма на языке программирования Julia:

```
function make_step(axis, point, round-
ing_mode = RoundUp)
   search_point = copy(point)
   rounding_up = rounding_mode == RoundUp

   if abs(search_point[axis] -
- round(search_point[axis])) <= 10^-10
      search_point[axis] =
   = search_point[axis] +
   +(rounding_up ? 1 : -1)
   else
      search_point[axis] =
   round(search_point[axis], round-
   ing_mode)
   end

   transformed_point = L(point)
   transformed_search_point =
= L(search_point)

   if minimum(transformed_search_point) <
< 0
      got_out_of_bounds = true

      plane_axis = argmin(trans-
   formed_search_point)
      plane_normal = zeros(length(ba-
   sis_indices))
      plane_normal[plane_axis] = 1.0

      search_point_intersection =
   =intersect_line_with_plane(
         transformed_search_point -
      transformed_point,
         transformed_point,
         plane_normal,
         [0.0, 0.0],
      )

      search_point =
   Linv(search_point_intersection)
   else
      got_out_of_bounds = false
   end

   return (search_point,
got_out_of_bounds)
end
```

Блок-схема данной процедуры изображена на рисунке 7.

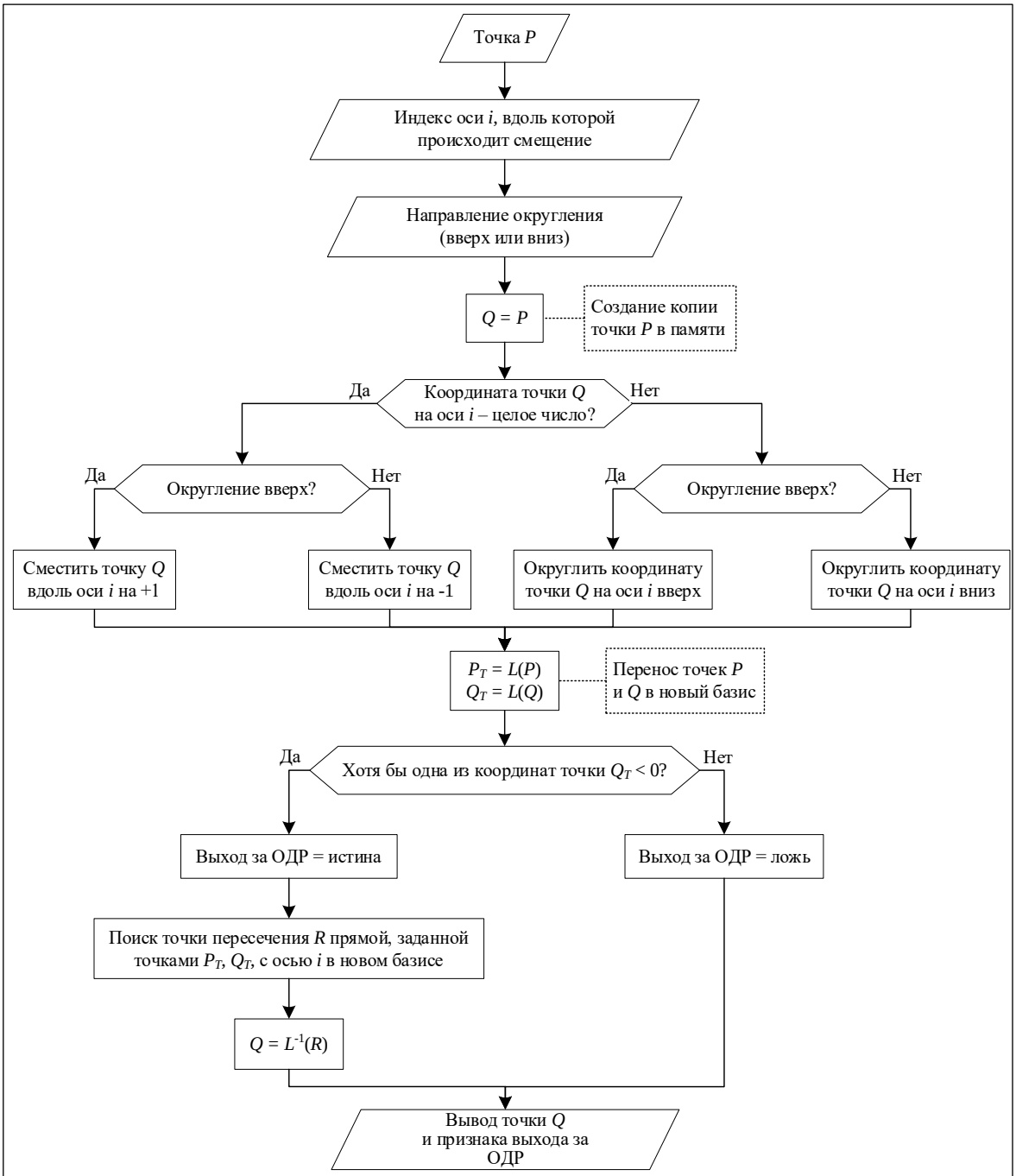


Рис. 7. Блок-схема алгоритма выполнения одного смещения точки

Fig. 7. Flow chart of the algorithm for a single step of the point

Наблюдение 1. Округление в сторону направления луча

Изменим ОДР, сместив и повернув второе ограничение, и аналогичным образом построим луч, направленный вглубь ОДР. Точка O' необязательно должна находиться внутри ОДР, поскольку через нее проводятся прямые, параллельные осям координат и пересекающиеся с лучом, направленным вглубь ОДР. Благодаря этому точка будут находиться внутри ОДР.

Когда луч направлен вниз и влево, округления вверх или вправо, как правило, не дают результата (за исключением, когда базисное ограничение параллельно оси координат).

Таким образом, если луч направлен влево, то есть первая координата вектора $\bar{v}$ отрица-

тельная, точку следует округлять вниз по оси x_1; если же первая координата вектора $\bar{v}$ положительная, соответствующую координату точки необходимо округлять вверх.

Аналогично для оси x_2 стоит руководствоваться знаком второй координаты вектора $\bar{v}$, определяющей луч. Так, если луч направлен вниз, то значение координаты x_2 у него отрицательное, следовательно, округлять координату точки необходимо также вниз. Если луч направлен вверх, то значение координаты x_2 у его вектора положительное и округлять координату точки необходимо вверх. Далее представлен процесс выполнения алгоритма поиска решения (рис. 8).

Исследование алгоритма при узкой ОДР

Изменим задачу так, чтобы ОДР была более узкой, а следовательно, целочисленное решение находилось бы дальше от оптимальной точки O. Из-за этого нужно будет выполнить больше шагов алгоритма, поэтапно округляя точку сначала по одной оси координат, а затем по другой. Используя уже представленный алгоритм, будем выполнять его итеративно до того момента, пока у точки P все координаты не окажутся целочисленными в исходном базисе. На рисунке 9 видно, что точка P как бы отскакивает от ограничений при выходе за пределы ОДР.

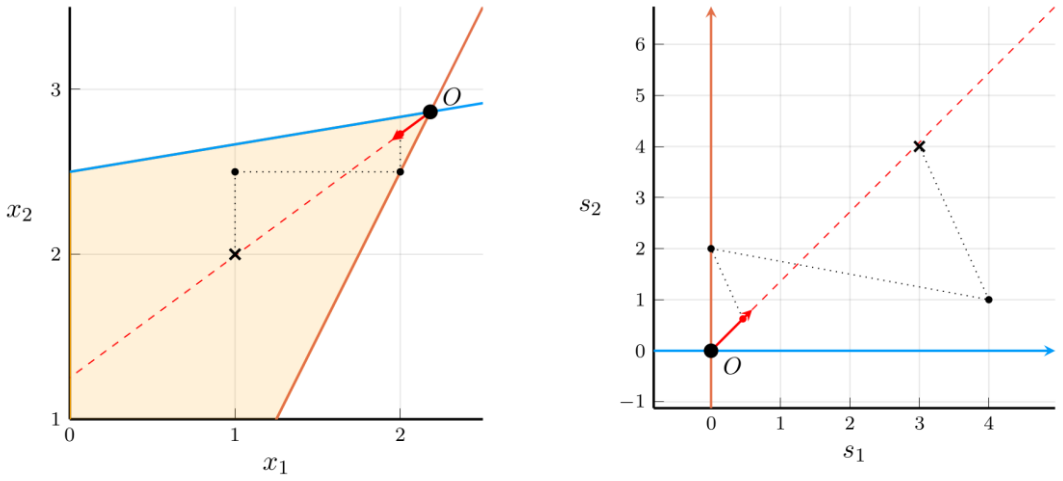


Рис. 8. Перемещение точки в процессе поиска решения

Fig. 8. Point movement during the search of a solution

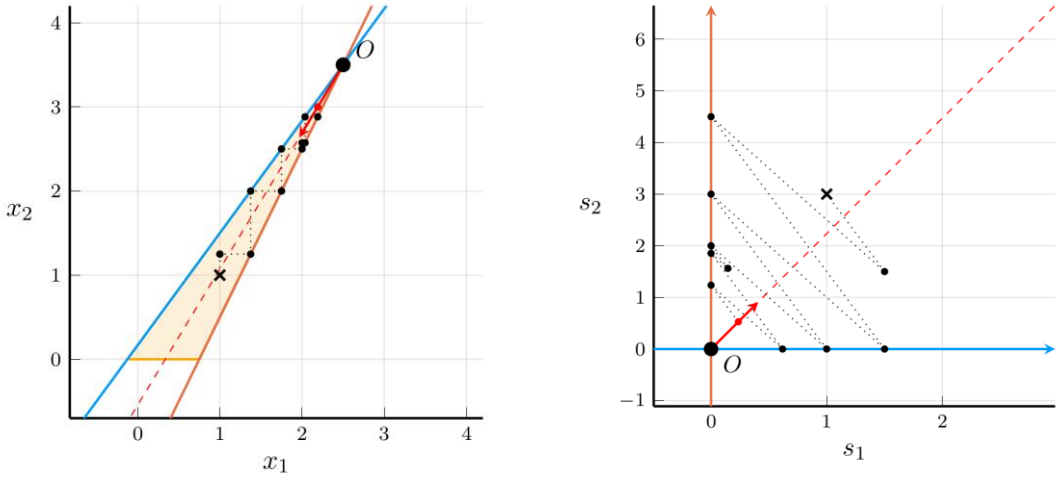


Рис. 9. Перемещение точки в процессе поиска решения для узкой ОДР

Fig. 9. Point movement during the search of a solution in the narrow feasible region

Исследование алгоритма при добавлении третьего ограничения

Введем третье ограничение, которое изменяло бы форму ОДР так, чтобы в ходе выполнения алгоритма точка, находящаяся внутри базиса между двумя ограничениями (s_1 и s_2), смогла выйти за пределы ОДР. В таком случае проверка на выход из ОДР и возврат точки обратно невозможны, и это приведет к тому, что алгоритм может не сойтись, хотя в текущем случае он находит решение (рис. 10).

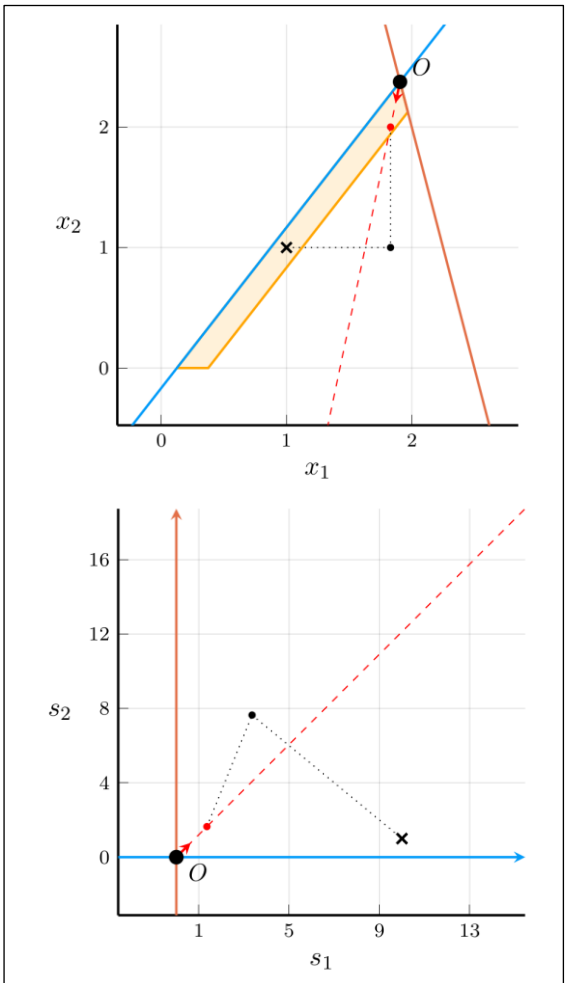


Рис. 10. Выход точки P за пределы ОДР при положительных координатах в базисе s_1s_2

Fig. 10. Point P leaves the feasible region while all of its coordinates in basis s1s2 are positive

Наблюдение 2. Определение последовательности округления координат

Используя вектор, задающий направление луча, можно определить последовательность осей координат, по которым стоит производить округление на следующих шагах. Дополняя наблюдение 1, определим веса $\overline{w}$ каждой из осей как соответствующие координаты нормализованного вектора $\hat{r}$, определяющего луч в исходном базисе:

$$\hat{r} = \frac{L^{-1}(\hat{v})}{\left|L^{-1}(\hat{v})\right|},$$

$$\overline{w} = \begin{pmatrix} |\hat{r}_1| \\ |\hat{r}_2| \end{pmatrix} \approx \begin{pmatrix} |-0.198| \\ |-0.980| \end{pmatrix} \approx \begin{pmatrix} 0.198 \\ 0.980 \end{pmatrix}.$$

Учитывая вектор $\overline{w}$, сначала нужно округлять по второй координате, а затем по первой, так как значение $|\hat{r}_2|$ больше $|\hat{r}_1|$. Аналогично можно определить направление округления, если использовать значения координат вектора $\hat{r}$ не по модулю, а по знаку:

$$\overline{d} = \begin{pmatrix} \hat{r}_1 \\ \hat{r}_2 \end{pmatrix} \approx \begin{pmatrix} -0.198 \\ -0.980 \end{pmatrix} \begin{matrix} \downarrow \\ \downarrow \end{matrix}.$$

Таким образом, вектор $\overline{d}$ указывает, что по оси x_1 (первый элемент вектора) округлять нужно вниз (то есть влево на графике), как и по оси x_2 (второй элемент вектора) (то есть вниз на графике). Например, если бы вектор $\hat{r}$ содержал значения с разными знаками, округлять стоило бы в направлении знака по соответствующим осям координат, поскольку именно в ту сторону направлен луч:

$$\overline{d} = \begin{pmatrix} \hat{r}_1 \\ \hat{r}_2 \\ \hat{r}_3 \end{pmatrix} \approx \begin{pmatrix} 0.37 \\ -0.65 \\ 0.76 \end{pmatrix} \begin{matrix} \uparrow \\ \downarrow \\ \uparrow \end{matrix}.$$

В таком случае округлять по осям x_1 и x_3 необходимо было бы вверх, а по оси x_2 – вниз.

Исследование алгоритма при изменении направления луча

Изменим ОДР, повернув ограничение s_2 так, чтобы луч смотрел не влево, а вправо. Тем самым продемонстрируем несостоятельность текущего способа построения луча, поскольку в таком случае алгоритм никогда не сойдется и целочисленное решение, лежащее внутри ОДР, не будет найдено (рис. 11).

Заключение

В статье был рассмотрен алгоритм поиска целочисленного решения, основанный на построении луча, смотрящего вглубь ОДР, и итеративном процессе округления координат

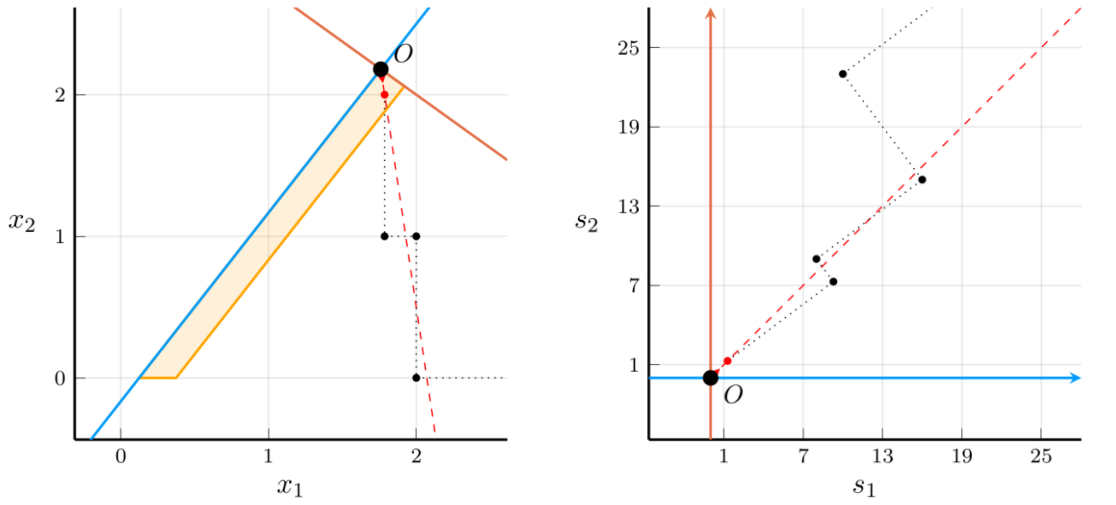


Рис. 11. Отдаление точки P от ОДР и расхождение алгоритма

Fig. 11. Point P moves away from the feasible region and the algorithm diverges

точки в направлении данного луча. Проведенные эксперименты показали работоспособность данного подхода, однако не всегда возможно построить луч, который смотрел бы вглубь ОДР. Кроме того, предложенный алгоритм может работать некорректно при введении дополнительных ограничений. Таким образом, в настоящий момент алгоритм подходит только для задач с двумя ограничениями.

Дальнейшие исследования могут быть направлены на модификацию алгоритма, чтобы он мог работать при большем числе ограничений. Также необходимо изменить способ построения луча. Например, его можно строить не среди ограничений, а между точками оптимального решения и начального базиса, что должно исключить возможность выхода точки за пределы ОДР.

Список литературы

1. Баусова З.И., Гамазина А Н., Дугина Ю.Н., Старикова А.Ю. Решение задач целочисленной линейной оптимизации // Вестн. ПензГУ. 2017. № 4. С. 117–121.

2. Арженовский С.В., Синявская Т.Г., Рудяга А.А. Решение задачи целочисленной оптимизации для фирмы // Учет и статистика. 2017. № 4. С. 36–43.

3. Легков К.Е., Нестеренко О.Е. Алгоритм формирования информационной структуры параллельных программ иерархической вычислительной системы // Наукоемкие технологии в космических исследованиях Земли. 2017. Т. 9. № 1. С. 52–59.

4. Гальмукова И.А. Линейное программирование в задачах экономики // Социально-экономические проблемы развития предпринимательства: региональный аспект: материалы VI Междунар. конф. 2017. С. 333–338.

5. Абдуахадов А.А. Краткая история развития математического программирования // Проблемы науки. 2021. № 6. С. 6–8.

6. Ямашкин Ю.В., Новокрещенова О.А. Системный подход к организации. Саранск, 2016. 195 с.

7. Матвеев Ю.Н., Иванов А.В. Использование метода покоординатного спуска для поиска решения задачи целочисленного программирования // Вестн. ТвГТУ. Сер. Технич. науки. 2023. Т. 17. № 1. С. 53–62.

8. Пинягина О.В. Метод покоординатного спуска для задачи рыночного равновесия с ценовыми группами // Учен. зап. Казан. ун-та. Сер. Физ.-матем. науки. 2018. Т. 160. № 4. С. 718–730.

9. Мельников Б.Ф., Мельникова Е.А. О классической версии метода ветвей и границ // Компьютерные инструменты в образовании. 2021. № 1. С. 21–44. doi: 10.32603/2071-2340-2021-1-21-45.

10. Смагин Б.И., Машин В.В. Критический анализ решения задачи целочисленного линейного программирования методом Гомори // Наука и образование. 2022. Т. 5. № 1. URL: http://opusmgau.ru/index.php/see/article/view/4520/4552 (дата обращения: 13.04.2023).

11. Якубова У.Ш., Парпиева Н.Т., Мирходжаева Н.Ш. Некоторые применения графического и симплексного методов решения задач линейного программирования // Бюллетень науки и практики. 2022. Т. 8. № 4. С. 490–498. doi: 10.33619/2414-2948/77/57.

12. Адельшин А.В., Кучин А.К. Исследование L-структуры многогранника смешанной задачи максимальной выполнимости // ПДМ. 2017. № 38. С. 110–118. doi: 10.17223/20710410/38/9.

13. Атутова Н.Д. Применение эвристических методов для поиска булевых функций с криптографическими характеристиками // Прикладная дискретная математика. Приложение. 2022. № 15. С. 18–21. doi: 10.17223/2226308X/15/5.

14. Максимова Н.Н., Колтунов Н.С. Поиск оптимального кольцевого маршрута на карте г. Благовещенска с использованием алгоритма имитации отжига // Вестн. АмГУ. 2020. № 91. С. 3–8. doi: 10.22250/jasu.1.

15. Хамхоева Ф.Я. Нейронные сети в экономическом анализе: плюсы и минусы // Norwegian J. of Development of the Int. Sci. 2020. Т. 4. № 51. С. 72–75.

16. Толок А.В., Толок Н.Б. Решение задач математического программирования функционально-воксельным методом // Проблемы управления. 2017. № 3. С. 37–42.

Software & Systems doi: 10.15827/0236-235X.142.539-550 2023, vol. 36, no. 4, pp. 539–550

An integer programming solution: Iterative rounding of coordinates

Aleksey V. Ivanov
Yury N. Matveev

For citation
Ivanov, A.V., Matveev, Yu.N. (2023) ‘An integer programming solution: Iterative rounding of coordinates’, *Software & Systems*, 36(4), pp. 539–550 (in Russ.). doi: 10.15827/0236-235X.142.539-550
Article info
Received: 24.04.2023 After revision: 02.05.2023 Accepted: 17.05.2023

Abstract. The article proposes an algorithm for finding an integer solution using the idea of rounding coordinates of an optimal non-integer solution point and constructing a half-line directed deep into an acceptable solution area. The proposed algorithm is based on an iterative process of rounding coordinates of a point in the direction of a constructed half-line. The study educed that moving towards a half-line without going through all possible options simplifies the algorithm and avoids branching, which distinguishes this approach compared to other currently existing open methods, such as the method of clipping and the method of branches and boundaries. The aim of the work is to develop and study an algorithm for finding an optimal integer solution using the idea of rounding coordinates of a point of an optimal non-integer solution. During the study, the authors carried out description and experimental verification of this algorithm and the possibility of its application in different forms of an acceptable solution domain. The theoretical significance of the work is a development of a new algorithm that does not require performing a simplex-method at each stage, and uses a half-line instead of a plane at each algorithm step, which prevents an increase in problem spatial complexity compared to other methods. The study showed that the proposed algorithm has limitations; however, the main idea proved its operability, which is planned to be developed further.

Keywords: mathematical programming, linear programming, integer programming, optimization, algorithm, coordinate descent

References

1. Bausova, Z.I., Gamazina, A.N., Dugina, Yu.N., Starikova, A.Yu. (2017) ‘Solving integer linear optimization problems’, *Vestn. of Penza State University*, (4), pp. 117–121 (in Russ.).

2. Arzhenovskiy, S.V., Sinyavskaya, T.G., Rudyaga, A.A. (2017) ‘Solving an integer optimization problem for a company’, *Accounting and Statistics*, (4), pp. 36–43 (in Russ.).

3. Legkov, K.E., Nesterenko, O.E. (2017) ‘The algorithm for forming the information structure of parallel programs of a hierarchical computing system’, *High Tech in Earth Space Research*, 9 (1), pp. 52–59 (in Russ.).

4. Galmukova, I.A. (2017) ‘Linear programming in economics’, Proc. 6th Int. Conf. *Social and Economic Problems of Entrepreneurship Development: Regional Aspect*, pp. 333–338 (in Russ.).

5. Abduakhadov, A.A. 'Brief history of mathematical programming development', *Problems of Sci*., 2021, (6), pp. 6–8 (in Russ.).

6. Yamashkin, Yu.V., Novokreshchenova, O.A. (2016) *A Systematic Approach to the Organization*. Saransk, 195 p. (in Russ.).

7. Matveev, Yu.N., Ivanov, A.V. (2023) 'Usage of coordinate descent algorithm in search of a solution to an integer programming problem', *Vestn. of TvSTU. Ser. Tech. Sci.*, 17(1), pp. 53–62 (in Russ.).

8. Pinyagina, O.V. (2018) 'A coordinate descent method for market equilibrium problems with price groups', *Uchenye Zapiski Kazanskogo Universiteta. Ser. Fiziko-Matemat. Nauki*, 160 (4), pp. 718–730 (in Russ.).

9. Melnikov, B.F., Melnikova, E.A. (2021) 'On the classical version of the branch and bound method', *Computer Tools in Education*, (1), pp. 21–44 (in Russ.). doi: 10.32603/2071-2340-2021-1-21-45.

10. Smagin, B.I., Mashin, V.V. (2022) 'Critical analysis of the solution of the problem of integer linear programming by the Gomori method', *Sci. and Education*, 5 (1), available at: http://opusmgau.ru/index.php/see/article/view/4520/4552 (accessed April 13, 2023) (in Russ.).

11. Yakubova, U.Sh., Parpieva, N.T., Mirkhodzhaeva, N.Sh. (2022) 'Some applications of graphical and simplex methods for solving linear programming problems', *Bull. of Sci. and Pract*., 8 (4), pp. 490–498 (in Russ.). doi: 10.33619/2414-2948/77/57.

12. Adelshin, A.V., Kuchin, A.K. (2017) 'Analysis of L-structure of polyhedron in the partial max sat problem', *Applied Discrete Math.*, (38), pp. 110–118 (in Russ.). doi: 10.17223/20710410/38/9.

13. Atutova, N.D. (2022) 'Application of heuristic methods to search for Boolean functions with good cryptographic characteristics', *Applied Discrete Math. Supplement*, (15), pp. 18–21 (in Russ.). doi: 10.17223/2226308X/15/5.

14. Maksimova, N.N., Koltunov, N.S. (2020) 'Search for an optimal ring route on the Blagoveshchensk map using the annealing simulation algorithm', *Messenger AmSU*, (91), pp. 3–8 (in Russ.). doi: 10.22250/jasu.1.

15. Khamkhoeva, F.Ya. (2020) 'Neural networks in economic analysis: Pros and cons', *Norwegian J. of Development of the Int. Sci*., (51)4, pp. 72–75 (in Russ.).

16. Tolok, A.V., Tolok, N.B. (2017) 'Mathematical programming problems solving by functional voxel method', *Problemy Upravleniya*, (3), pp. 37–42 (in Russ.).

Авторы
Иванов Алексей Викторович [1], аспирант, lexuzieel@gmail.com
Матвеев Юрий Николаевич [1], д.т.н., профессор, matveev4700@mail.ru

[1] Тверской государственный технический университет, г. Тверь, 170026, Россия

Authors
Aleksey V. Ivanov [1], Postgraduate Student, lexuzieel@gmail.com
Yury N. Matveev [1], Dr.Sc. (Engineering), Professor, matveev4700@mail.ru

[1] Tver State Technical University, Tver, 170026, Russian Federation