HARDWARE AND SOFTWARE COMPLEX FOR AUTOMATIC CONTROL OF THE DRYING AGENT PREPARATION PROCESS WITH PRESET TEMPERATURE AND HUMIDITY PARAMETER

A.V. KAYCHENOV[1], Cand. Sc., V.V. ERESHCHENKO[1], Sen. Lect.
V.V. YATSENKO[1], Cand. Sc., I.G. BLAGOVESHCHENSKY[2], Dr. Sc.

[1]Murmansk State Technical University,
13, Sportivnaya str., Murmansk, 183010, Russian Federation,
e-mail: kaychenovav@mstu.edu.ru
[2]Moscow State University of Food Production,
11, Volokolamsk hwy, Moscow, 125080, Russian Federation, e-mail: igblagov@mgupp.ru

The article describes a hardware and software complex for automatic control of the drying agent preparation process with specified temperature and humidity parameters. The hardware includes a single-board computer, temperature and humidity sensor, control boards for the refrigerator and flaps, communication boards. The software component of the complex includes modules for polling sensors, controlling the refrigeration machine, calculating the degree of flap opening, and managing data exchange. A mathematical description of the processes of heating, cooling, dehumidification, humidification and mixing of air flows is presented, which served as the basis for the visualization module in the form of an interactive I-d diagram that allows calculating the parameters of the processes of humidification, heating, cooling and mixing of the drying agent. The results of experiments on adiabatic humidification and heating of one section of a small-sized drying unit UPOR-M are presented.

Keywords: hardware and software complex, automatic control, drying agent, mathematical description, interactive I-d-diagram.

Поступила в редакцию/received: 21.07.2022; после рецензирования/revised: 05.09.2022;
принята/accepted: 12.09.2022

УДК 004.021

ИСПОЛЬЗОВАНИЕ МЕТОДА ПОКООРДИНАТНОГО СПУСКА ДЛЯ ПОИСКА РЕШЕНИЯ ЗАДАЧИ ЦЕЛОЧИСЛЕННОГО ПРОГРАММИРОВАНИЯ

Ю.Н. МАТВЕЕВ, д-р техн. наук, А.В. ИВАНОВ, аспирант

Тверской государственный технический университет,
170026, Тверь, наб. Аф. Никитина, 22, e-mail: matveev4700@mail.ru

Предложен алгоритм решения задачи целочисленного программирования способом, аналогичным методу покоординатного спуска. Альтернативный алгоритм позволяет избежать трудоемкого поиска точки с помощью симплекс-метода на каждой итерации. Приведены подробное описание алгоритма, демонстрация его работы на примере задачи оптимизации с двумя параметрами, текущие ограничения

предлагаемого алгоритма, а также рассуждения по поводу возможного их устранения в дальнейших работах.

Ключевые слова: дискретная оптимизация, математическое программирование, линейное программирование, целочисленное программирование, оптимизация, алгоритм, градиентный метод, покоординатный спуск.

DOI: 10.46573/2658-5030-2023-1-53-62

ВВЕДЕНИЕ

Задачи целочисленного программирования (ЦП) являются частным случаем задач линейного программирования (ЛП) и находят свое применение уже достаточно долгое время, поскольку данный вид оптимизационных задач позволяет учитывать дискретность, то есть неделимость объектов. Впервые способы решения подобных задач были предложены в XX в. Джорджом Данцигом и Леонидом Канторовичем [1] и открыли возможность поиска оптимальных решений задач. Из-за того, что такие задачи оптимизации, как задача оптимального размещения [2] или задача с логическими и ресурсными ограничениями [3], требуют соблюдения целочисленности значений при поиске решения, то нахождение наиболее оптимального алгоритма решения задачи ЦП является достаточно актуальной.

В настоящее время существует два направления развития алгоритмов ЦП. Представители первого занимаются теоретическим исследованием уже существующих алгоритмов, как правило, основанных на использовании принципа релаксации, в частности алгоритмов метода отсечения и метода ветвей и границ. В рамках второго создают новые алгоритмы или разновидности уже имеющихся задач ЦП, призванных ускорять процесс решения. К данному направлению относят современные итеративные эвристические алгоритмы, такие как поиск с восхождением к вершине [4] и алгоритм имитации отжига [5], являющийся примером метода Монте-Карло [6]. Кроме того, в последнее время все более популярными становятся эвристические алгоритмы, применяющие генетический подход [7], и принципы машинного обучения [8].

К сожалению, в настоящий момент не наблюдается развития точных алгоритмов решения задач ЦП. Вместо этого используются разработанные в начале становления данной области методы, такие как метод ветвей и границ [9], а также метод отсечений и их модификации. Однако указанные методы были разработаны достаточно давно и обладают высокой вычислительной сложностью из-за итеративного подхода.

ПОСТАНОВКА ЗАДАЧИ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Задача ЛП в общем виде представляется в виде системы линейных уравнений и вектора коэффициентов целевой функции $\bar{z}$:

$$M = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} & b_1 \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} & b_m \end{pmatrix}; \qquad \bar{z} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix},$$

где m – число ограничений; n – число осей координат.

В свою очередь левая часть матрицы M – это матрица A с коэффициентами уравнений ограничений, определяющих область допустимых решений, а правая – вектор $\bar{b}$ свободных членов каждого из m уравнений:

$$A_{m,n} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}; \bar{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

Определим необходимым условием для работы алгоритма поиска целочисленного решения наличие оптимального *нецелочисленного* решения в виде точки $\bar{p}$ с координатами $(p_1 \quad p_2 \quad \cdots \quad p_n)$ в n-мерном пространстве (рис. 1). Данная точка может быть найдена любым доступным способом решения обычной задачи линейного программирования (например, симплекс-методом [10]).

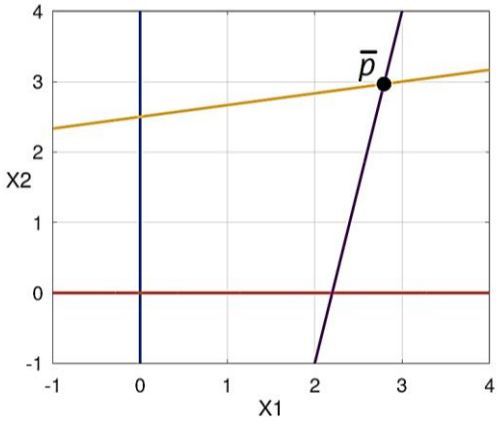

Рис. 1. Определение области допустимых решений и оптимальное решение в точке $\bar{p}$

ОПИСАНИЕ АЛГОРИТМА

Суть алгоритма заключается во введении дополнительных целочисленных ограничений по аналогии с методом отсечений и перемещении точки оптимального решения через перерасчет одной из ее координат таким образом, чтобы она лежала на линии целочисленного ограничения. Как мы видим, перемещение точки решения аналогично перемещению, используемому в градиентном методе покоординатного спуска (по аналогии с методом Гаусса – Зейделя). Сам алгоритм состоит из следующих шагов:

1) выбора оси координат, на которой будет производиться поиск;
2) введения целочисленных ограничений для выбранной оси;
3) нахождения точек пересечения с целочисленными ограничениями;
4) отсеивания полученных точек;
5) выбор «опорной» точки;
6) определения направления движения и перемещения оптимального решения;
7) возврата к шагу 1, если точка оптимального решения не имеет все целочисленные координаты или находится за пределами области допустимых решений (ОДР), в противном случае алгоритм завершен.

Рассмотри каждый из шагов подробно.

Выбор оси координат для поиска. Первым шагом алгоритма является выбор оси координат, по которой будут производиться поиск и перемещение точки, определяющей текущее временное решение. В настоящий момент оси координат выбираются по очереди в произвольном порядке (например, сначала ось x_1, затем ось x_2 и так далее). В последующих работах будут рассмотрены особенности выбора осей координат и влияние данного выбора на сходимость алгоритма.

Введение целочисленных ограничений. После того как была выбрана одна из осей координат, в нее вводятся два целочисленных ограничения в окрестности текущей

точки оптимального решения $\bar{p}$ (рис. 2). Для этого значение координаты точки $\bar{p}$ на выбранной оси округляется как в вверх, так и вниз.

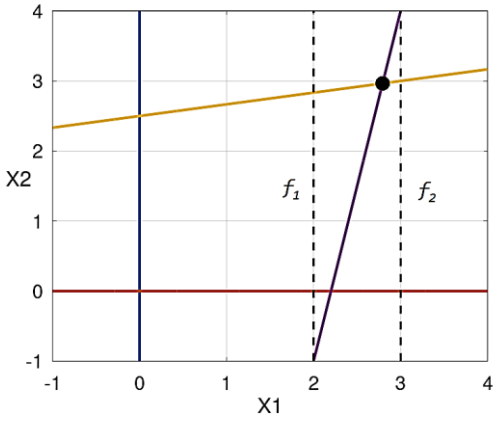

Рис. 2. Введение целочисленных ограничений в окрестности точки $\bar{p}$ при выборе оси x_1

Нахождение точек пересечения с целочисленными ограничениями. Поскольку столбцы в матрице A соответствуют осям координат, то для того, чтобы найти набор значений координат на оси, соответствующих каждому ограничению из системы в указанной точке, достаточно решить уравнение

$$\bar{p}_i = \left(\bar{b} - A_{subset} \cdot \bar{p}_{subset}^T\right) \oslash \bar{a}. \quad (1)$$

По сути, уравнение (1) – выражение одной из координат при фиксированных остальных по аналогии с уравнением $y = kx + b \rightarrow x = -\frac{b-y}{k}$, если его записать в векторной форме. Можно заметить, что уравнение $x = -\frac{b-y}{k}$ очень схоже с уравнением (1). В нем вектор $\bar{p}_i$ – это $-x$, вектор $\bar{b}$ – это b, y – это $A_{subset} \cdot \bar{p}_{subset}^T$, а множитель k – это вектор $\bar{a}$. Здесь и далее символом «$\oslash$» будем обозначать поэлементное деление векторов. A_{subset} и $\bar{p}_{subset}$ – измененные версии матрицы A и вектора $\bar{p}$ с исключенными столбцом и элементом вектора, соответствующие оси координат i, на которой осуществляется поиск, а вектор $\bar{a}$ – это непосредственно сам исключенный столбец из матрицы A:

$$A_{subset} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,i-1} & a_{1,i+1} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,i-1} & a_{2,i+1} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n-1} & a_{m,i+1} & \cdots & a_{m,n} \end{pmatrix};$$

$$\bar{p}_{subset} = (p_1 \quad p_2 \quad \ldots \quad p_{i-1} \quad p_{i+1} \quad \ldots \quad p_n);$$

$$\bar{a} = \begin{pmatrix} a_{1,i} \\ a_{2,i} \\ \vdots \\ a_{m,i} \end{pmatrix}.$$

Результатом выражения является вектор $\bar{p}_i$, содержащий m элементов (по числу ограничений системы) со значениями координат для оси, на которой производится поиск. Добавив недостающие столбцы в их соответствующие места (руководствуясь номером оси i), получим матрицу P, которая представляет собой набор точек на

пересечении с каждым из ограничений при значениях координат на фиксированных осях, соответствующих вектору $\bar{p}_{subset}$:

$$P = \begin{pmatrix} p_1 & p_2 & \cdots & p_{i,1} & \cdots & p_n \\ p_1 & p_2 & \cdots & p_{i,2} & \cdots & p_n \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ p_1 & p_2 & \cdots & p_{i,m} & \cdots & p_n \end{pmatrix}. \quad (2)$$

Например, пусть задана система ограничений:

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

а точка оптимального нецелочисленного решения:

$$\bar{p} = (2{,}793 \quad \mathbf{2{,}966}).$$

Тогда матрица A и вектор $\bar{b}$ будут выглядеть следующим образом:

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

Жирным шрифтом выделен элемент вектора и столбец матрицы, соответствующие второй оси координат ($i = 2$), на которой будет осуществляться поиск (в данном случае это x_2 на двухмерной координатной плоскости). *Все остальные* оси (в данном случае только x_1) считаются зафиксированными, то есть значения координат на этих осях не изменяются во время расчетов. Вместо этого для каждой из фиксированных осей *по очереди* производится округление как вверх, так и вниз. Таким образом, строятся дополнительные ограничения, пролегающие через ближайшие целочисленные значения координат каждой фиксированной оси. В соответствии с уравнением (1) определим A_{subset}, $\bar{a}$ и $\bar{p}_{subset}$:

$$A_{subset} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ -5 \end{pmatrix}; \qquad \bar{a} = \begin{pmatrix} 0 \\ 1 \\ -6 \\ 1 \end{pmatrix}; \qquad \bar{p}_{subset} = (2{,}966).$$

Затем рассчитаем наборы координат $\bar{p}_i$ при $i = 2$ (поиск на оси x_2) для двух возможных вариантов, а именно с округлением **только элемента оси** i вверх и вниз (для оси x_2 это 2,966).

$$\bar{p}_2^{ceil} = \left(\bar{b} - A_{subset} \cdot \lceil \bar{p}_{subset} \rceil^T\right) \oslash \bar{a} =$$

$$= \frac{\left(\begin{pmatrix} 0 \\ 0 \\ -15 \\ -11 \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \\ 1 \\ -5 \end{pmatrix} \cdot (3)\right)}{\begin{pmatrix} 0 \\ 1 \\ -6 \\ 1 \end{pmatrix}} = \frac{\begin{pmatrix} -3 \\ 0 \\ -18 \\ 4 \end{pmatrix}}{\begin{pmatrix} 0 \\ 1 \\ -6 \\ 1 \end{pmatrix}} = \begin{pmatrix} -Inf \\ 0 \\ 3 \\ 4 \end{pmatrix};$$

$$\bar{p}_2^{floor} = \left(\bar{b} - A_{subset} \cdot \lfloor \bar{p}_{subset} \rfloor^T\right) \oslash \bar{a} =$$

$$= \left(\begin{pmatrix} 0 \\ 0 \\ -15 \\ -11 \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \\ 1 \\ -5 \end{pmatrix} \cdot (2)\right) \oslash \begin{pmatrix} 0 \\ 1 \\ -6 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ 0 \\ -17 \\ -1 \end{pmatrix} \oslash \begin{pmatrix} 0 \\ 1 \\ -6 \\ 1 \end{pmatrix} = \begin{pmatrix} -Inf \\ 0 \\ 2{,}83 \\ -1 \end{pmatrix}.$$

В результате получим матрицы со структурой, определенной в выражении (2):

$$P_2^{ceil} = \begin{pmatrix} 3 & -Inf \\ 3 & 0 \\ 3 & 3 \\ 3 & 4 \end{pmatrix}; \quad P_2^{floor} = \begin{pmatrix} 2 & -Inf \\ 2 & 0 \\ 2 & 2{,}83 \\ 2 & -1 \end{pmatrix}.$$

Объединив полученные матрицы по строкам и исключив несуществующие решения (включающие бесконечность), получим набор возможных «опорных» точек для выбора дальнейшего направления движения при перемещении точки $\bar{p}$ (рис. 3):

$$P_2 = \begin{pmatrix} 3 & 0 \\ 3 & 3 \\ 3 & 4 \\ 2 & 0 \\ 2 & 2{,}83 \\ 2 & -1 \end{pmatrix}.$$

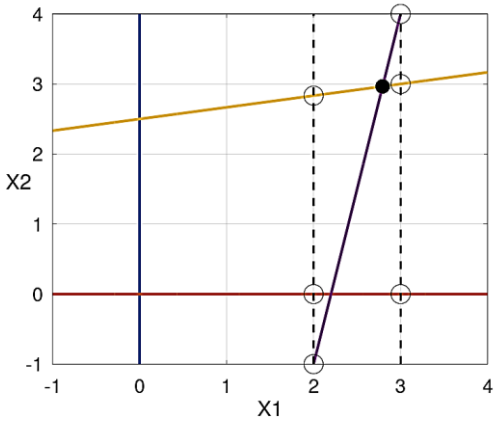

Рис. 3. Полученные точки пересечения с целочисленными ограничениями

Отсеивание полученных точек. После того как были найдены точки пересечения всех ограничений с каждым из целочисленных ограничений, выбираем только те точки, которые входят в ОДР. Для этого перемножим матрицы P_2 и A^T и получим в результате матрицу B', которая представляет собой решение относительно каждой точки, определенной в P_2. Каждая строчка B' – это вектор $\bar{b}_k$ относительно каждой конкретной точки $\bar{p}_k$, где k – это номер точки из матрицы P_2:

$$B' = P_2 \times A^T = \begin{pmatrix} 3 & 0 & 3 & -15 \\ 3 & 3 & -15 & -12 \\ 3 & 4 & -21 & -11 \\ 2 & 0 & 2 & -10 \\ 2 & 2{,}83 & -15 & -7{,}16 \\ 2 & -1 & 8 & -11 \end{pmatrix}.$$

Узнать, какие точки из P_2 находятся в ОДР, можно, построив матрицу B, в которой каждая строчка является вектором $\bar{b}$, а число строк равно числу строк матрицы B':

$$B = \begin{pmatrix} 0 & 0 & -15 & -11 \\ 0 & 0 & -15 & -11 \\ 0 & 0 & -15 & -11 \\ 0 & 0 & -15 & -11 \\ 0 & 0 & -15 & -11 \\ 0 & 0 & -15 & -11 \end{pmatrix}.$$

Если произвести поэлементное сравнение $B' \geq B$, то получим бинарную матрицу T:

$$T = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ 1 & 0 & 1 & 1 \end{pmatrix}.$$

В матрице T номера строк, содержащих все единицы, соответствуют точкам из P_2, находящихся в ОДР (рис. 4): $P'_2 = \begin{pmatrix} 2 & 0 \\ 2 & 2{,}83 \end{pmatrix}$. Переопределим $P = P'_2$, поскольку в дальнейшем будут использоваться только точки, входящие в ОДР.

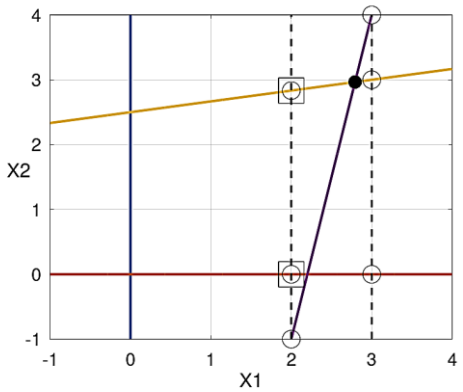

Рис. 4. Отсеянные точки, входящие в ОДР (отмечены квадратами)

Выбор «опорной» точки. Для каждой из точек, входящих в ОДР, рассчитываем значение целевой функции. В матричном виде это выглядит следующим образом:

$$\bar{z} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{pmatrix}; \quad P = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,m} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{k,1} & p_{k,2} & \cdots & p_{k,m} \end{pmatrix},$$

где $\bar{z}$ – вектор коэффициентов целевой функции из m элементов; матрица P – это набор из k точек с m координатами, находящиеся в ОДР.

Перемножив между собой вектор коэффициентов $\bar{z}$ и матрицу P, получим вектор $\bar{j}$ из k элементов, где каждый элемент – это значение целевой функции в соответствующей точке:

$$\bar{J} = \bar{z} \times P^T = (j_1 \quad j_2 \quad \cdots \quad j_k).$$

Определение направления движения и перемещение точки. В зависимости от того, какая задача решается (максимизации или минимизации), выбирается наибольший или наименьший элемент из вектора $\bar{J}$ и соответствующая ему точка s (рис. 5).

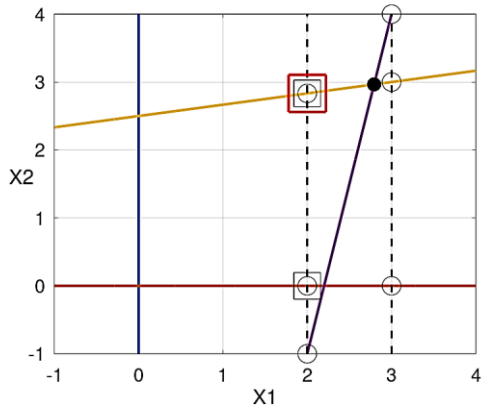

Рис. 5. Точка с наибольшим значением целевой функции

Затем для текущей оси i, на которой производится поиск, высчитывается разность между значениями координат текущей и опорной точками. Если текущую и опорную точки определить в виде векторов

$$\bar{p} = (p_1 \quad p_2 \quad \dots \quad p_n); \qquad \bar{p}_{basis} = (P_{s1} \quad P_{s2} \quad \dots \quad P_{sn}),$$

тогда разность их координат можно рассчитать как

$$\Delta_i = P_{si} - p_i,$$

где i – номер текущей оси, на которой осуществляется поиск; s – номер точки, которая была выбрана в качестве базисной.

Для текущего примера

$$\bar{p} = (2{,}793 \quad 2{,}966); P = \begin{pmatrix} 2 & 0 \\ 2 & 2{,}83 \end{pmatrix} \rightarrow s = 2;$$

$$\Delta_i = P_{si} - p_i = 2{,}83 - 2{,}966 = -0{,}136; \;\; i = 2.$$

В зависимости от того, является ли значение Δ_i положительным или отрицательным, происходит перемещение точки вдоль оси i либо вверх, либо вниз.

Если значение Δ_i отрицательное, то это означает то, что следующая ближайшая точка с целочисленной координатой на оси i находится *ниже*, то есть необходимо *уменьшить* значение целевой функции в задачи максимизации для того, чтобы изменить координату на оси i на целочисленную. Справедливо и обратное: если значение Δ_i положительное, то следующая ближайшая точка с целочисленным значением координаты на оси i находится *выше*, а это означает, что нужно *увеличить* значение целевой функции.

Изменение значения координаты точки на оси i происходит путем округления *вниз*, если значение Δ_i отрицательное, либо *вверх*, если значение Δ_i положительное. В случае если значение Δ_i равно нулю, то это означает, что координата точки на оси i уже является целым числом. Используя ранний пример, пересчет точки текущего решения осуществляем следующим образом (рис. 6):

$$\Delta_i = -0{,}136 < 0 \rightarrow \bar{p} = (2{,}793 \quad 2); \;\; i = 2.$$

После изменения точки текущего решения проводится проверка на нахождение ее в ОДР. Если точка расположена в ОДР и все координаты точки являются целочисленными, то алгоритм считается завершенным, а целочисленное оптимальное решение – найденным. В противном случае происходит возврат на первый шаг алгоритма, он выполняется заново для нового текущего решения и другой оси координат (рис. 7).

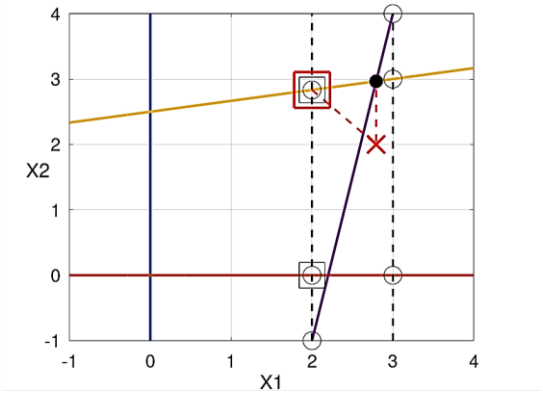

Рис. 6. Изменение значения координаты i текущего решения

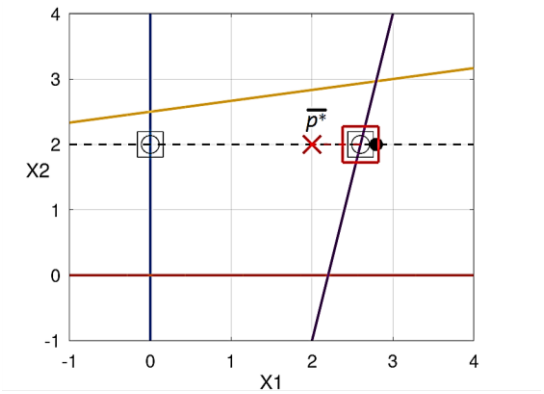

Рис. 7. Нахождение оптимального целочисленного решения $\overline{p^*}$

ЗАКЛЮЧЕНИЕ

Предложенный метод решения задач ЦП во многих случаях позволяет избежать решения многомерной задачи с использованием симплекс-метода на каждой итерации алгоритма. В отличие от таких методов, как метод ветвей и границ и метод отсечений, предложенный алгоритм позволяет потенциально ускорить время поиска решения задачи ЦП. Однако в ходе проведенных экспериментов было отмечено, что во время выполнения вышеописанного алгоритма при перемещении точки решения и ее выходе за ОДР этот алгоритм не всегда может сойтись, а сходимость является важным требованием точности алгоритма решения задачи ЦП. Отсюда следует сделать вывод о том, что проверку выхода за ОДР точки решения нужно производить на каждом шаге алгоритма и предотвращать этот выход на этапе определения того, куда переместить точку решения. Данное требование приводит к значительному усложнению логики алгоритма, что, на наш взгляд, не является целесообразным. И хотя алгоритм, предложенный в данной работе, не является универсальным и с помощью него нельзя найти решение любой задачи ЦП, однако, на наш взгляд, очевиден тот факт, что имеет смысл поиск альтернативного подхода к поиску решения, поэтому в дальнейших статьях мы собираемся продолжить работу в данном направлении.

ЛИТЕРАТУРА

1. Данциг Дж.Б. Линейное программирование, его применения и обобщения / под ред. Н.Н. Воробьева. М.: Прогресс. 1966. 600 с.

2. Васильев И.Л., Климентова К.Б. Метод ветвей и отсечений для задачи размещения с предпочтениями клиентов // *Дискретный анализ и исследование операций*. 2009. Т. 16. № 2. С. 21–41.

3. Колоколов Л.А., Семерханова Е.Я. Оптимизация системы производственных услуг в условиях межфирменного взаимодействия // *Вестник СибГУТИ*. 2014. № 3 (27). С. 13–22.

4. Russell S.J., Norvig P. Artificial Intelligence: A Modern Approach / Upper Saddle River. New Jercey: Prentice Hall. 2003. 1132 p.

5. Джонс М.Т. Программирование искусственного интеллекта в приложениях. М.: ДМК Пресс. 2004. 312 с.

6. Соболь И.М. Метод Монте-Карло. М.: Наука. 1968. 64 с.

7. Гладков Л.А., Курейчик В.В, Курейчик В.М., Сороколетов П.В. Биоинспирированные методы в оптимизации. М.: Физматлит. 2009. 384 с.

8. Джонс М.Т. Программирование искусственного интеллекта в приложениях. М.: ДМК Пресс. 2004. 312 с.

9. Васильев И.Л., Климентова К.Б. Метод ветвей и отсечений для задачи размещения с предпочтениями клиентов // *Дискретный анализ и исследование операций*. 2009. Т. 16. № 2. С. 21–41.

10. Таха Хемди А. Введение в исследование операций / перевод с англ. 7-е изд. М.: Вильямс. 2005. 912 с.

Для цитирования: Матвеев Ю.Н., Иванов А.В. Использование метода покоординатного спуска для поиска решения задачи целочисленного программирования // Вестник Тверского государственного технического университета. Серия «Технические науки». 2023. № 1 (17). С. 53–62.

USAGE OF COORDINATE DESCENT ALGORITHM IN SEARCH OF A SOLUTION TO AN INTEGER PROGRAMMING PROBLEM

Y.N. MATVEEV, Dr. Sc., A.V. IVANOV, Postgraduate

Tver State Technical University,
22, Af. Nikitin emb., Tver, 170026, Russian Federation, e-mail: matveev4700@mail.ru

The article proposes an algorithm for solving the integer programming problem in a way similar to the coordinate descent method. An alternative algorithm avoids the time-consuming search for a point using the simplex method at each iteration. A detailed description of the algorithm and a demonstration of its operation is given on the example of an optimization problem with two parameters. The current limitations of the proposed algorithm are given, as well as arguments about their possible use in future works.

Keywords: discrete optimization, mathematical programming, linear programming, integer programming, optimization, algorithm, gradient method, coordinate descent.

Поступила в редакцию/received: 16.08.2022; после рецензирования/revised: 05.09.2022; принята/accepted: 12.09.2022