2.3.1.

А.В. Иванов, Ю.Н. Матвеев д-р техн. наук

ФГБОУ ВО «Тверской государственный технический университет»,
кафедра «Электронные вычислительные машины»,
Тверь, lexuzieel@gmail.com

АЛГОРИТМ РЕШЕНИЯ ЗАДАЧ ЦЕЛОЧИСЛЕННОГО ПРОГРАММИРОВАНИЯ С ПРИМЕНЕНИЕМ ИТЕРАТИВНОГО ОКРУГЛЕНИЯ КООРДИНАТ

В работе описан алгоритм решения задачи целочисленного программирования с применением идеи покоординатного спуска. Описывается способ определения начальной точки и метод ее поэтапного смещения вглубь области допустимого решения вплоть до нахождения целочисленного решения. Предложенный способ позволяет избежать выхода точки за область допустимого решения на каждом шаге алгоритма.

Ключевые слова: *линейное программирование, целочисленное программирование, оптимизация, алгоритм, покоординатный спуск.*

Введение. Целочисленное программирование (ЦП) [1] продолжает оставаться востребованным методом решения задач оптимизации. Оно играет важную роль в оптимизации различных процессов на производстве и решении сложных задач распределения ресурсов во многих отраслях. Одной из главных причин актуальности ЦП является способность учитывать дискретные переменные при решении. Одними из широко используемых методов решения задач ЦП до сих пор являются метод Гомори или метод ветвей и границ [2], а также метод отсечений [3]. Несмотря их популярность, у данных методов существуют недостатки. В частности у данных методов достаточно высокая вычислительная сложность – особенно при большой размерности задачи.

В данной статье мы описываем развитие идеи итеративного округления координат [4] применительно к решению задач целочисленного программирования. Мы предлагаем измененный метод построения луча внутри области допустимых решений (ОДР), а также представляем идею того, как можно избежать выхода точки из ОДР во время выполнения алгоритма.

Постановка задачи. В предлагаемом алгоритме используется точка оптимального решения *O* и точка с минимальным значением целевой функции *S*, которые всегда находятся в ОДР. Поскольку ОДР является выпуклым многоугольником, то можно быть уверенными, что на отрезке между точками *S* и *O* все точки находятся также внутри ОДР. Используя данный факт можно построить луч, который бы всегда смотрел вглубь ОДР, вне зависимости от угла между смежными ограничениями в точке *O*.

Если точку *O* можно найти любым известным методом поиска решения задачи линейного программирования (нецелочисленного), например, таким как симплекс-метод [5], то для нахождения точки *S* необходимо найти начальный допустимый базис. Существует несколько известных способов определения начального базиса, таких как М-метод [6] и двухфазный симплекс-метод [7]. Данные методы относятся к методам искусственного базиса и позволяют определить начальный базис за счет ввода искусственных переменных. Допустим, даны следующие ограничения и целевая функция:

$$f(x) = 3x_1 + 2x_2 \to max$$

$$\begin{cases} -x_1 + 6x_2 \le 15 \\ 4x_1 - 2x_2 \le 3 \\ x_1 + x_2 \ge 0.5 \\ x_1 \ge 0 \\ x_2 \ge 0 \end{cases}$$

Используя двухфазный симплекс метод можно найти сразу обе точки: точку оптимального нецелочисленного решения O и точку, соответствующую начальному базису (минимальному значению целевой функции) S (рис. 1). Таким образом, будут получены точки, через которые можно провести луч $\hat{v}$, направленный вглубь ОДР и «смотрящий» на точку S. Данный луч необходим для определения начальной точки алгоритма.

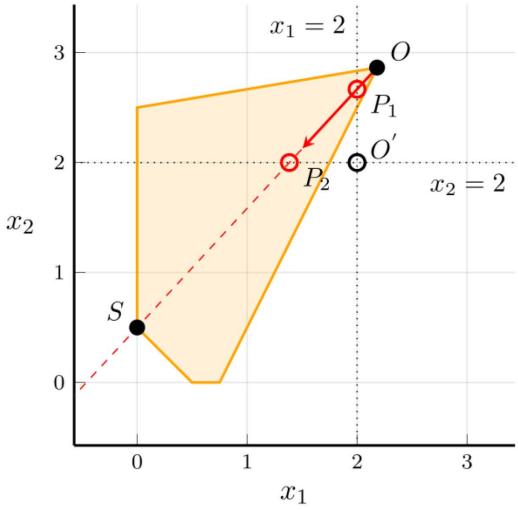

Рис. 1 – Точка оптимального решения O и начальная точка S, а также луч, смотрящий вглубь ОДР с потенциальными начальными точками на нем

Определение начальной точки. Для того чтобы найти начальную точку, из которой будет происходить дальнейший поиск, необходимо округлить координаты точки O вниз (в задаче максимизации):

$$O' = \lfloor O \rfloor = (2,2).$$

После этого через полученную точку проведем секущие плоскости $x_1 = 2$, $x_2 = 2$, параллельные осям координат (рис. 1) и найдём точки пересечения P_i между плоскостями и лучом. Таким образом, получим набор потенциальных начальных точек внутри ОДР, в каждой из которых одна из координат является целочисленной.

Затем необходимо выбрать «наилучшую» точку — это можно сделать, рассчитав значения целевой функции $f(x)$ для каждой точки P_i. Для примера возьмем целевую функцию: $f(x) = 3x_1 + 2x_2$, тогда можно найти следующие значения целевой функции:

$$f(P_1) = 3 \cdot 2 + 2 \cdot \frac{8}{3} = \frac{34}{3} \approx 11.333332$$

$$f(P_2) = 3 \cdot \frac{18}{13} + 2 \cdot 2 = \frac{106}{13} \approx 8.1538461$$

Значение целевой функции для P_1 больше, следовательно, данная точка будет начальной. Поэтому из нее будет осуществляться поиск целочисленного решения. Обозначим ее как $P = P_1$.

Нахождение следующей точки. Основная часть алгоритма заключается в поиске следующей точки, у которой хотя бы одна из координат также является целочисленной. Для этого, через текущую точку P, лежащую на луче $\hat{v}$ проведем прямую $\hat{q}$, параллельную оси, вдоль которой происходит смещение, и разделим ее на две части. На первой половине будут находиться точки, со значениями координат меньше, чем у точки P вдоль этой же оси, а на второй половине — точки, со значениями координат больше, чем у точки P. Тем самым получится луч, который направлен в сторону уменьшения значения целевой функции (т.е. вглубь ОДР для задачи максимизации). После этого можно найти пересечение всех ограничений с прямой $\hat{q}$ и выбрать только те точки, которые находятся в первой части. Точка с наименьшей разностью координаты с P вдоль текущей оси будет находиться на границе ОДР (исходя из признака выпуклости ОДР). Таким образом, полученная точка будет всегда гарантированно находиться на границе ОДР (рис. 2).

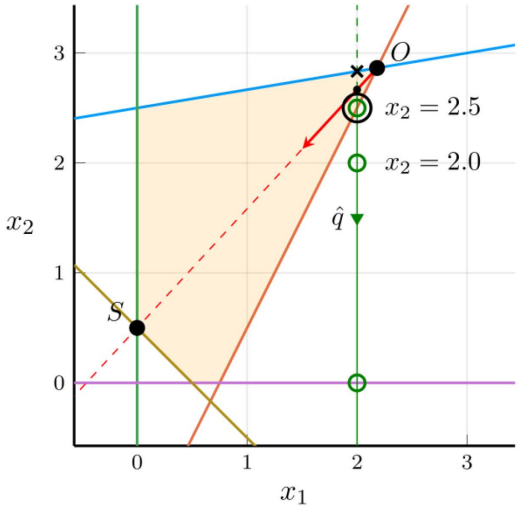

Рис. 2 – Выбор следующей ближайшей точки

Здесь зеленым кружком отмечены точки, находящиеся в первой половине прямой, в то время как точка, отмеченная крестиком — во второй. Зеленая точка, обведенная черным кружком, является следующей точкой алгоритма. Направление движения в данном случае было выбрано по направлению луча $\hat{v}$.

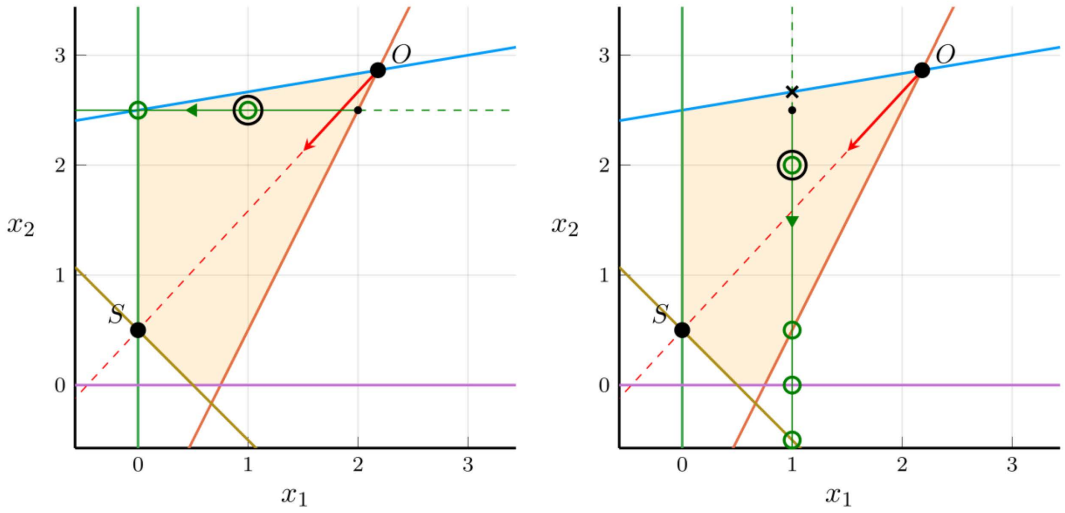

Рис. 3 – Итеративный процесс выполнения алгоритма

Также необходимо отметить, что помимо точек пересечения прямой $\hat{q}$ с ограничивающими плоскостями стоит дополнительно ввести точку P со значением координаты вдоль текущей оси, округленным вниз (точка со значением $x_2 = 2$ на рис. 2). Это позволит избежать случая, когда ближайшая точка на границе ОДР находится слишком далеко и избежать «перескока» целочисленной точки внутри ОДР.

Аналогичным образом можно построить луч $\hat{q}$ из новой точки P (рис. 3). В данном случае новая точка сначала придет в точку (1, 2.5), а затем в точку (1, 2), которая является целочисленным решением данной задачи, следовательно, целочисленное решение найдено и алгоритм выполнен.

Заключение. Подводя итог, можно сказать, что модифицированный алгоритм работает лучше, чем предложенный нами в предыдущей работе. Предложенный алгоритм гарантирует нахождение внутри ОДР на каждой итерации, а также позволяет осуществить поэтапное смещение вглубь ОДР вдоль осей координат по аналогии с методом покоординатного спуска.

Благодаря тому, что для нахождения точек вводятся дополнительные прямые, а не плоскости это потенциально может дать значительный выигрыш в производительности по сравнению с методами, которые вводят дополнительные ограничивающие плоскости.

В дальнейших работах мы планируем рассмотреть применение предложенного алгоритма для задач в большей мерности. Кроме того более детального описания заслуживает способ определения выбора последовательности осей координат.

Список литературы

1. *Баусова З.И., Гамазина А Н., Дугина Ю.Н., Старикова А.Ю.* Решение задач целочисленной линейной оптимизации // Вестник ПензГУ. 2017. Т. 20. № 4.
2. *Смагин Б.И., Машин В.В.* Критический анализ решения задачи целочисленного линейного программирования методом Гомори // Наука и образование. 2022. № 1.
3. *Мельников Б.Ф., Мельникова Е.А.* О классической версии метода ветвей и границ // Компьютерные инструменты в образовании. 2021. № 1.
4. *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. Т. 17. № 1.
5. *Хемди А. Таха.* Глава 3. Симплекс-метод // Введение в исследование операций. — 7-е изд. — М.: «Вильямс», 2007. — С. 95—141.
6. *Васильев Ф.П., Иваницкий А.Ю.* Линейное программирование (3-е издание). 2008. Факториал. 356 с.
7. *Добрынина Н.Ф., Конина А.А.* Двухфазный симплекс-метод. Математическое и компьютерное моделирование естественно-научных и социальных проблем: сб. ст. IX Междунар. (2015): 76.