Дата последней сборки документа:
20 февраля 2022 г. в 21:12

Общие и специальные вопросы оптимизации

Подвойский А.О.

Краткое содержание

Содержание

1. Полезные ссылки

https://github.com/ceandrade/brkga_mip_feasibility

2. Методы решения задач линейного программирования

Задачи линейного программирования относятся к подклассу задач выпуклого программирования [3, стр. 57].

2.1. Симплекс-метод Данцига

Замечание

В худшем случае симплекс-метод работает за экспоненциальное время, но в целом на практике обычно он работает очень быстро, и многочисленные эксперименты и исследования метода подтвердили полиномиальное время работы [3, стр. 61]

Для оценки вычислительной сложности симплекс-метода Данцига Спилман и Тенг предложили использовать *сглаженный анализ*. Сглаженный анализ – вероятностный анализ алгоритма, при котором изучается работа алгоритма при незначительном случайном возмущении конкретных входных данных, а затем ищется зависимость производительности алгоритма от размера входа и от среднеквадратичного отклоненеия возмощений.

Так вот Спилман и Тенг показали, что симплекс-метод имеет *полиномиальную сглаженную сложность*[1]. Эти результаты означают, что хотя и существуют задачи, на которых симплекс-метод будет работать *экспоненциально* долго, но если исходные данные таких задач (коэффициенты целевой функции и ограничений) подвергнуть незначительному изменению, то с достаточно высокой вероятностью симплекс-метод на возмущенной задаче уже будет работать за *полиномиальное время* [3, стр. 62].

В конце 1970-х годов обнаружили, что алгоритмы, в общем случае решающие задачи линейного программирования за полиномиальное время, все-таки существуют. Все такие полиномиальные алгоритмы – *методы внутренней точки* (первый метод внутреней точки с полиномиальной сложностью – метод Кармаркара) и *метод эллипсоидов* – отличались от симплекс-метода геометрическим подходом.

В течение 50 лет оставался открытым вопрос, существует ли полиномиальный алгоритм, который работает подобно симплекс-методу, перебирая только вершины (угловые точки) допустимого множества задачи. Ответ на этот вопрос дали Келнер и Спилман в 2006 году: они представили рандомизированный симплекс-метод с *полиномиальным временем работы*.

Замечание

В современных оптимизационных пакетах задачи линейного программирования средней размерности (вплость до сотен тысяч и даже миллиона переменных или ограничений) решаются методами внутренней точки.

Постановка задачи. Найти максимум функции

$$f(x) = \sum_{j=1}^{n} c_j x_j \leftarrow c^T x$$

при ограничениях

$$\sum_{j=1}^{n} a_{ij} x_j = b_i,\ i = 1, \ldots, m\ (m < n) \leftarrow Ax = b,$$

$$x_j \geqslant 0,\ j = 1, \ldots, n.$$

[1]В том же 2006 году оценка симплекс-метода Спилмана и Тенга была улучшена. Р. Вершининым. Он показал, что ожидаемое время работы симплекс-метода на незначительно измененных входных данных является полиномом от логарифма количества ограничений n

Такая постановка называется *канонической*, а искомое решение $x^* = (x_1^*, \ldots, x_n^*)^T$ – *оптимальным.*

Замечания:

- Максимизируемая функция и ограничения <u>линейны</u> по $x_j,\ j = 1, \ldots, n,$
- Задача содержит ограничения на неотрицательность переменных, присутствие которых диктуется процедурой симплекс-метода. Если по физической постановке задачи какая-либо переменная, например, x_n, неограничена по знаку, то ее можно представить в виде $x_n = x_{n+1} - x_{n+2}$, где $x_{n+1}, x_{n+2} \geqslant 0$,
- В ограничениях $\sum\limits_{j=1}^{n} a_{ij}x_j = b_i,\ i = 1, \ldots, m\ (m < n)$ будем считать переменные $b_i \geqslant 0,\ i = 1, \ldots, m.$

Стратегия метода Данцига решения описаной задачи основана на особенностях постановки этой задачи. Множество

$$X = \{x \mid \sum_{i}^{n} a_{ij}x_j = b_i,\ i = 1, \ldots, m,\ x \in \mathbb{R}^n,\ x_j \geqslant 0,\ j = 1, \ldots, n\}$$

допустимых решений задачи – есть выпуклое множество, которое геометрически представляет собой *выпуклый политоп*[2], имеющий конечное число *крайних точек.*

Крайней точкой выпуклого множества X называется точка $x \in X$, которая не может быть выражена в виде выпуклой комбинации других точек $y \in X, x \neq y$.

Стратегия решения задачи симплекс-методом состоит в направленном переборе базисных решений, определяющих крайние точки политопа. Направленность перебора предполагает следующую организацию вычислительного процесса [2]:

1. Нахождение базисного решения (метод Гаусса-Жордана, переход к M-задаче),
2. Переход от одного базисного решения к другому таким образом, чтобы обеспечить улучшение целевой функции (другими словами, переход от одной *вершины политопа* к другой в направлении улучшения целевой функции).

3. Методы решения задач линейного целочисленного программирования

3.1. Метод ветвей и границ

Постановка задачи (Mixed Integer Linear Programming, MILP)

$$f(x) = \sum_{j=1}^{n} c_j x_j \to \max$$

при ограничениях

$$\sum_{j=1}^{n} a_{ij}x_j \geqslant b_i,\ i = 1, \ldots, m,$$
$$x_j \geqslant 0,\ x \in \mathbb{Z},\ j = 1, \ldots, n.$$

[2]Политоп – подмножество Евклидова пространства, представимое объединением симплексов (n-мерное обобщение треугольника)

Замечание

Описанная задача является задачей линейного целочисленного программирования. Ограничения, связанные с целочисленностью, могут быть наложены не на все переменные, а лишь на их часть

Стратегия поиска

Для корня дерева ветвей-и-границ (branch-and-bound tree) описанная задача решается *симплекс-методом* без учета ограничений на целочисленность (т.е. в релаксированной постановке). Считается, что она имеет решение. На полученном оптимальном решении $x^{0*} = (x_1^{0*}, \ldots, x_n^{0*})$ вычисляется значение *целевой функции* $f(x^{0*})$.

Если решение x^{0*} является целочисленным, то поставленная задача решена. Если решение x^{0*} оказывается нецелочисленным, то значение $f(x^{0*})$ (полученное для решаемой задачи в релаксированной постановке) является *верхней границей* (потому что мы решаем задачу на максимум) возможных оптимальных значений $f(x)$ на целочисленных решениях.

При нецелочисленном решении дальнейшая процедура решения задачи состоит в ее ветвлении на две подзадачи. Целью этого ветвления является разбиение множества допустимых решений на два подмножества путем построения дополнительных ограничений таким образом, чтобы исключить нецелочисленную точку x^{0*} и сделать решение по крайней мере одной из задач целочисленным по одной выбранной координате x_k.

Координатой x_k может быть [2, стр. 339]:

1. Нецелочисленная координата с наименьшим или наибольшим индексом.
2. Нецелочисленная координата с наименьшей или наибольшей дробной частью.
3. Нецелочисленная координата, которой соответствует наибольший коэффициент в целевой функции.
4. Нецелочисленная координата, выбранная на основани приоритетов, определяемых физическим содержанием задачи.

Для построения дополнительных ограничений округляем нецелочисленное решение вниз и исследуем область значений левее, т.е. $x_k \leqslant \lfloor x_k^{0*} \rfloor$, и округляем вверх и исследуем область правее, т.е. $\lceil x_k^{0*} \rceil \leqslant x_k$.

Построение дополнительных ограничений позволило исключить из рассмотрения оптимальное нецелочисленное решение x^{0*} и обеспечить целочисленность значений координаты x_k.

Задачи ЗЛП-1 и ЗЛП-2 записываются в виде, изображенном на рис. 2.

Задачи ЗЛП-1 и ЗЛП-2 решаются самостоятельно симплекс-методом *без учета ограничений на целочисленность* координта $x_j,\ j = 1, \ldots, n$. Вычисляются значения функции $f(x)$ на оптимальных решениях обеих задач. Если ни одна из них не имеет целочисленного решения, то выбрирается задача для приоритетного дальнейшего ветвления по установленному правилу: например, приоритетному ветвлению подлежит та задача, в которой значение $f(x)$ на оптимальном *нецелочисленном* решении максимально.

Пусть $f(x^{1*}) > f(x^{2*})$, тогда задача ЗЛП-1 первой ветвится на ЗЛП-3 и ЗЛП-4, которые решаются симплекс-методом *без учета требований на целочисленность* с последующим анализом решений [2, стр. 340]. Если ни одна из задач ЗЛП-3 и ЗЛП-4 не имеет целочисленного решения, приступают к ветвлению задачи ЗЛП-2.

Процесс ветвления продолжается до тех пор, пока не будет получено в одной из ветвей целочисленное решение. Пусть задача ЗЛП-4 имеет целочисленное решение. Обозначим $\underline{f}$ – значение

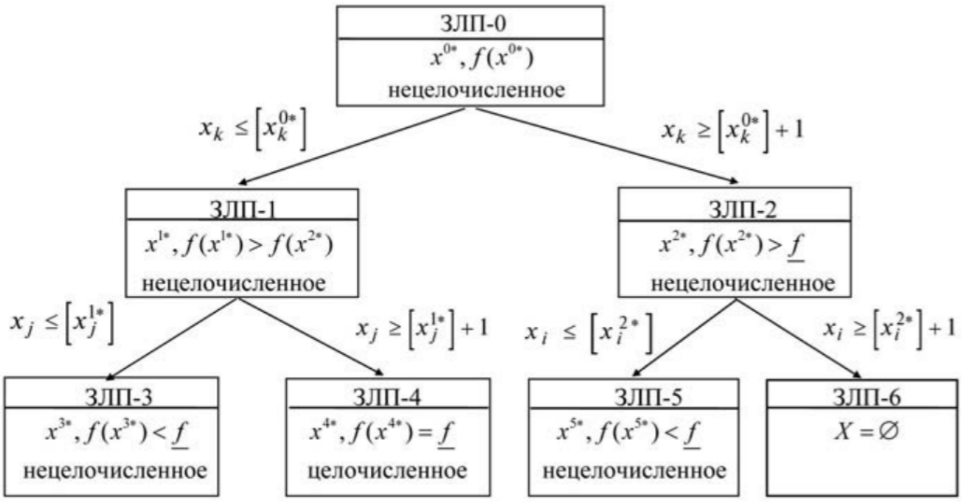

Рис. 1. Пример дерева ветвей-и-границ

ЗЛП - 1

$$f(x) = \sum_{j=1}^{n} c_j x_j \to \max;$$
$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad i = 1, \ldots, m;$$
$$x_k \le [x_k^{0*}];$$
$$x_j \ge 0, \quad j = 1, \ldots, n;$$

ЗЛП - 2

$$f(x) = \sum_{j=1}^{n} c_j x_j \to \max;$$
$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad i = 1, \ldots, m;$$
$$x_k \ge [x_k^{0*}] + 1;$$
$$x_j \ge 0, \quad j = 1, \ldots, n.$$

Рис. 2. Подзадачи корневого узла дерева ветвей-и-границ

функции на первом целочисленном решении: $\underline{f} = f(x^{4*})$. Соответсвующее целочисленное решение включается в множество $\bar{X}^*$ возможных оптимальных решений исходной задачи.

После того, как найдено первое целочисленное решение, вопрос о дальнейшем ветвлении других задач решается на основании сравнения значений $f(x^{k*})$ на оптимальных <u>нецелочисленных</u> решениях в оставшихся ветвях со значением $\underline{f}$.

Если $f(x^{k*}) \leqslant \underline{f}$ для всех оставшихся k, то расчет закончен. Решениями исходной задачи являются те целочисленные решения x^{k*}, для которых $f(x^{k*}) = \underline{f}$.

Если $f(x^{k*}) > \underline{f}$, то соответствующая этому номеру k задача ветвится далее. Так, на рис. 1 имеем $f(x^{2*}) > \underline{f}$ и $f(x^{3*}) < \underline{f}$. Задача ЗЛП-2 подлежит ветвлению на ЗЛП-5, ЗЛП-6, а задача ЗЛП-3 не подлежит. Задача ЗЛП-6 не имеет решения, так как множество допустимых решений пустое, и далее не рассматривается. Задача ЗЛП-5 имеет нецелочисленное решение $x^{5*}, f(x^{5*})$. Если $f(x^{5*}) < \underline{f}$, то решение задачи закончено и $x^* = x^{4*}, f(x^*) = \underline{f}$. В противном случае задача ЗЛП-5 ветвится дальше.

Если в одной из задач получено целочисленное решение, то ее ветвление далее не прозводится. Если соответствующее значение целевой функции $\geqslant \underline{f}$, решение считается принадлежащим множеству X^* возможных оптимальных решений исходной задачи.

Если значение целевой функции $< \underline{f}$, целочисленное решение не включается в множество X^*.

Таким образом, ветвление какой-либо задачи заканчивается, если выполняется одно из условий [2]:

1. решение целочисленное,
2. значение целевой функции данной задачи $\leqslant \underline{f}$,

3. множество допустимых решений пустое.

Если ветвление всех задач закончено, то в множестве X^* выбирается решение (решения), которому соответствует наибольшее значение целевой функции. Оно является решением исходной задачи. Если множество X^* пустое, то исходная задача не имеет решения.

3.2. Первичные эвристики в SCIP

Алгоритм ветвей-и-границ представляет собой *полную* (complete) процедуру. Это означает, что алгоритм гарантирует оптимальное решение каждой проблемы за конечное время. Однако, это очень дорогой в вычислительном смысле метод и в худшем случае временные издержки будут экспоненйиально зависеть от размерности задачи.

В противоположенность первичные эвристики относятся к *неполным* (incomplete) методам. То есть они пытаются найти допустимое решение приемлемого качества за небольшой промежуток времени. Но они не гарантируют оптимальности решения.

Первичные эвристики в SCIP условно могут быть разделены на 4 категории [1]:

- Эвристики округления (rounding heuristics) пытаются округлять значения переменных релаксированного решения таким образом, чтобы полученное округленное решение оставалось допустимым,
- Глубокие эвристики (diving heurisctics) начинают с текущего релаксированного решения задачи и итеративно фиксируют целочисленные значения целочисленных переменных и решают текующую задачу заново,
- Целевые глубокие эвристики (objective diving heuristics) похожи на глубокие эвристики, но вместо фиксации переменных, они изменяют значения коэффициентов в целевой функции,
- Улучшающие эвристики (improvenment heuristics) исследуют одно и несколько допустимых решений и пытаются построить такое решение, которому отвечает более низкое (в случае задачи минимизации) или более высокое (в случае задачи максимизации) значение целевой функции.

3.2.1. Эвристики округления

RENS

4. Общие положения постановки частично-целочисленного линейного программирования

Задача линейного программирования в частично-целочисленной постановке (Mixed Integer Linear Program, MILP, MIP) записывается в форме

$$\begin{aligned} \min c^T x&,\\ Ax = b&,\\ x \geqslant 0&,\\ x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I}& \end{aligned}$$

Задача, когда все переменные являются целочисленными, называется задачей линейного программирования в чистой целочисленной постановке (Pure Integer Linear Program, ILP, IP).

Если все переменные принимаются значения из множества $\{0, 1\}$, то задача называется задачей линейного программирования 0-1 (0-1 linear program).

Включение целочисленных переменных в постановку задачи расширяет возможности моделирования.

Задачи линейного программирования могут быть решены за полиномиальное время *методами внутренней точки* (метод эллипсоида, алгоритм Кармаркара).

Задачи целочисленного программирования относятся к классу NP-трудных:

- на текущий момент не известны алгоритмы, способные решить этот такого рода задачи за полиномиальное время,
- И, вообще говоря, есть мало шансов, что такие алгоритмы когда-нибудь будут найдены.

Релаксированное решение можно получить, сняв ограничения на целочисленность

$$\begin{aligned} \min c^T x, \\ Ax = b, \\ x \geqslant 0. \end{aligned}$$

Замечание

Задачу линейного программирования в частично-целочисленной постановке нельзя решить, просто перейдя от решения задачи в релаксированной постановке с последующем округлением переменных

Например, оптимальным решением задачи линейного программирования в чистой целочисленной постановке будет

$$\begin{aligned} \max x + y \\ -2x + 2y \geqslant 1, \\ -8x + 10y \leqslant 13, \\ x, y \geqslant 0, \\ x, y \in \mathbb{Z} \end{aligned}$$

вектор $(x, y) = (1, 2)$, которому отвечает целевая функция со значением 3.

А релаксированным оптимальным решением будет вектор $(x, y) = (4, 4.5)$ со значением целевой функции 9.5.

Не существует прямого способа перейти от релаксированного решения к целочисленному.

Предполагается, что переменные ограничены, т.е. имеют нижнюю и верхнюю границы.

Через P_0 обозначим рассматриваемую постановку задачи, а через $LP(P_0)$ релаксированное решение задачи P_0. Если в оптимальном релаксированном решении $LP(P_0)$ все целочисленные переменные принимают целочисленные значения, то это решение будет решением исходной задачи P_0.

В противном случае для целочисленной переменной x_j, которая принимает нецелочисленное значение β_j, $\beta_j \notin \mathbb{Z}$ в оптимальном релаксированном решении $LP(P_0)$, определяются подзадачи

$$\begin{aligned} P_1 := P_0 \wedge x_j \leqslant \lfloor \beta_j \rfloor, \\ P_2 := P_0 \wedge x_j \geqslant \lceil \beta_j \rceil. \end{aligned}$$

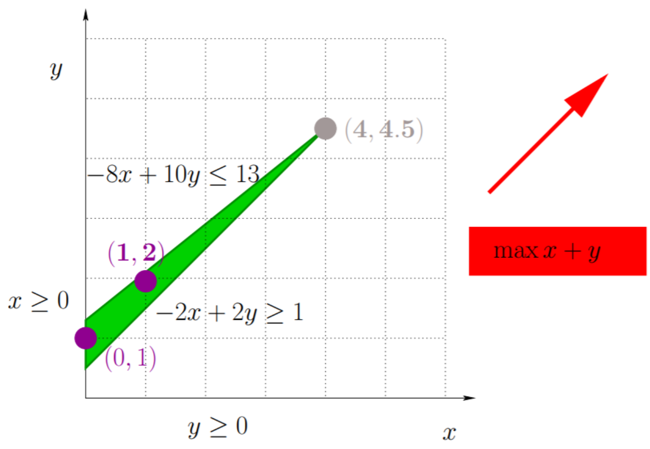

Рис. 3. Связь релаксированной и целочисленной постановок задачи

Тогда физичным решением исходной задачи будет

$$feasibleSols(P_0) = feasibleSols(P_1) \cup feasibleSols(P_2).$$

Список иллюстраций

Список литературы

1. *Achterberg T.* Constraint Integer Programming, 2007
2. *Пантлеев А. В., Летова Т.А,* Методы оптимизации в примерах и задачах. – СПб.: Издательство «Лань», 2015. – 512 с.
3. *Вороноцова Е.А.* Выпуклая оптимизация. – М.: МФТИ, 2021. – 364 с.
4. *Бурков А.* Машинное обучение без лишних слов. – СПб.: Питер, 2020. – 192 с.
5. *Бизли Д.* Python. Подробный справочник. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 864 с.