ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 24 за 2024 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Т.В. Сивакова, В.А. Судаков,
В.С. Шимко

Исследование методов решения задач смешанного целочисленного линейного программирования

Статья доступна по лицензии
Creative Commons Attribution 4.0 International

Рекомендуемая форма библиографической ссылки: Сивакова Т.В., Судаков В.А., Шимко В.С. Исследование методов решения задач смешанного целочисленного линейного программирования // Препринты ИПМ им. М.В.Келдыша. 2024. № 24. 18 с. https://doi.org/10.20948/prepr-2024-24
https://library.keldysh.ru/preprint.asp?id=2024-24

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В. Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

Т.В. Сивакова, В.А. Судаков, В.С. Шимко

Исследование методов решения задач смешанного целочисленного линейного программирования

Москва – 2024

Сивакова Т.В., Судаков В.А., Шимко В.С.

Исследование методов решения задач смешанного целочисленного линейного программирования

В работе рассматриваются математические методы и программное обеспечение, предназначенные для решения оптимизационных задач с линейными целевыми функциями и ограничениями в условиях дополнительных ограничений на целочисленность переменных. Изложены основные алгоритмы комбинаторной оптимизации, и проведён сравнительный анализ актуальных пакетов и решателей для решения задач смешанного целочисленного линейного программирования на языке Python.

Ключевые слова: смешанное целочисленное линейное программирование, программное обеспечение, алгоритм, оптимизация, целевая функция

Tatyana Vladimirovna Sivakova, Vladimir Anatolyevich Sudakov, Vasiliy Sergeyevich Shimko

Research of methods for solving mixed integer linear programming problems

The paper discusses mathematical methods and software designed to solve optimization problems with linear objective functions and constraints, subject to additional restrictions on the integrality of variables. The main algorithms for combinatorial optimization are outlined and a comparative analysis of current packages and solvers for solving mixed integer linear programming problems in Python is carried out.

Key words: mixed integer linear programming, software, algorithm, optimization, target function

Введение

Одним из важнейших классов задач математического программирования являются задачи смешанного целочисленного линейного программирования (СЦЛП). Целый ряд практических задач оптимизации [1] сводится к этому типу задач, например, к полностью целочисленным задачам линейного программирования сводится большинство известных комбинаторных задач. Доказано, что задачи СЦЛП относятся к классу NP–полных задач [2]. Практическое значение данного понятия заключается в сложности решения с вычислительной точки зрения, т.к. время вычисления растёт экспоненциально с увеличением размерности задачи. Из этого можно сделать вывод, что нецелесообразно выстраивать точные алгоритмы. В связи с этим часто прибегают к использованию приближенных эвристических и метаэвристических методов, особенно для решения практических задач большой размерности [2]. Выделяют две категории, имеющие часть схожих характеристик, общих алгоритмов для задач СЦЛП: алгоритмы отсекающей плоскости, которые представляют собой развитие симплекс-алгоритма, и комбинаторные алгоритмы, которые основаны на рациональном переборе совокупности возможных решений [3].

Общая задача СЦЛП трудна по существу, и для ее решения разработано большое количество алгоритмов. Некоторые из этих алгоритмов особенно эффективны для определенных классов задач СЦЛП, но не существует универсального алгоритма, который хорошо работал бы на практике для большого числа задач (скажем, с несколькими десятками ограничений и переменных). Ряд этих алгоритмов представлен ниже:

1. Приближенные алгоритмы. Эти алгоритмы порождают не оптимальное решение, а решения, отличающиеся от действительного оптимума заведомо не более чем на фиксированную долю этого оптимума.
2. Вероятностные алгоритмы. Иногда оказывается возможным построить алгоритмы, которые очень часто относительно некоторого вероятностного распределения на индивидуальных задачах работают неплохо – в смысле качества получаемых решений или затрачиваемого времени.
3. Частные случаи. Частные случаи NP–полной задачи могут быть простыми.
4. Экспоненциальные алгоритмы. Некоторые методы поиска с экспоненциальной сложностью (например, метод ветвей и границ) можно успешно применять к индивидуальным задачам разумного размера.
5. Локальный поиск. Одним из наиболее успешных подходов к решению трудных комбинаторных задач является дискретный аналог «метода спуска», известный как локальный поиск.

6. Эвристики и метаэвристики. Такие подходы, будучи неудовлетворительными с математической точки зрения, уверенно работают в практических ситуациях.

Каноническая формулировка задачи СЦЛП имеет следующий вид [4,5].

Требуется минимизировать следующую линейную форму:

$$z = \min_x cx,$$

для которой определены следующие неравенства:

$$Ax \leq b,$$
$$x \in \mathbb{R}^n, x_j \in \mathbb{Z}, \qquad \forall j \in I,$$

где A – матрица размерности $n \times m$, b и c – векторы размерности m и n соответственно, а $I \subseteq \{1, 2, 3 \dots, n\}$ – подмножество номеров целочисленных переменных.

Рассмотрим ряд методов для решения задач оптимизации более подробно.

Методы решения

Метод ветвей и границ относится к комбинаторным методам решения целочисленных задач и применим как к полностью, так и к смешанно целочисленным задачам, и формирует целое семейство алгоритмов. Суть метода ветвей и границ – в формировании дерева подзадач исходной задачи R и направленном частичном или полном переборе всех узлов этого дерева.

Работа алгоритма начинается с рассмотрения релаксированной исходной задачи линейного программирования (которая в большинстве случаев хорошо решается симплекс-методом, описанным ниже), т.е. задачи без ограничений на целочисленность. В результате решения релаксированной задачи находится нижняя граница $\check{c}$, а верхняя граница $\hat{c}$ – в ходе решения подзадач и проверки их решений алгоритмом на интегральность. Далее производится формирование двух или более подзадач путём создания дополнительных ограничений, которые затем решаются аналогично исходной [4].

Ниже приведено подробное описание работы алгоритма [4].

Алгоритм ветвей и границ

Вход: Оптимизационная задача R

Выход: Оптимальное решение x^* и значение функции c^*, или отсутствие таковых, если нет решения

1. Инициализировать список подзадач $\mathcal{L} \coloneqq \{R\}$ и верхнюю границу $\hat{c} := \infty$.
2. Если $\mathcal{L} = 0$, вернуть $x^* = \hat{x}$ и $c^* = \hat{c}$.
3. Выбрать подзадачу $Q \in \mathcal{L}$ и удалить Q из $\mathcal{L}$.
4. Решить релаксацию Q_{relax} подпроблемы Q. Если Q_{relax} пустая, то нижняя граница $\check{c} := \infty$. Иначе $\check{x}$ будет оптимальным решением Q_{relax}, а $\check{c}$ – его целевым значением.

5. Если $\check{c} \geq \hat{c}$, то перейти к шагу 2.
6. Если $\check{x}$ удовлетворяет условию интегральности для R, то $\hat{x} := \check{x}$, $\hat{c} := \check{c}$ и перейти к шагу 2.
7. Разбить Q на подзадачи $Q = Q_1 \cup \ldots \cup Q_k$, положить $\mathcal{L} := \mathcal{L} \cup \{Q_1 \cup \ldots \cup Q_k\}$ и перейти к шагу 2.

На эффективность метода ветвей и границ влияет множество решений, а точнее то, по каким правилам или с использованием каких методов и эвристик выбираются переменные, по которым производится ветвление, следующий узел дерева (подзадача из множества подзадач) для рассмотрения, по каким правилам узлы удаляются из дерева, а также правила создания сечений (в случае метода ветвей и сечений) [1]. Все эти элементы делают разработку и применение метода ветвей и границ для сложных задач весьма нетривиальными.

Метод ветвей и сечений является вариацией метода ветвей и границ. Применение метода отсекающих плоскостей позволяет редуцировать допустимые множества формируемых непрерывных релаксаций путём добавления новых линейных неравенств. Впервые этот подход был представлен Ральфом Гомори, однако на практике его алгоритм не применяется, т.к. он является весьма неэффективным из-за экспоненциального роста числа сечений, что очень затормаживает вычислительный процесс [4].

Чаще всего современные решатели комбинируют ветвление и сечение одним из двух следующих способов: сечение и ветвление (cut-and-branch) или ветвление и сечение (branch-and-cut) [4]. В первом случае релаксация исходной задачи подкрепляется секущими плоскостями, а затем производятся вычисления обычным методом ветвей и границ. Во втором случае производятся вычисления методом ветвей и границ, однако для подзадач могут быть использованы секущие плоскости в зависимости от разных правил и эвристик (например, сечения могут применяться только в определённой области, а не ко всему дереву).

Симплекс-метод – это метод последовательного перехода от одного базисного решения (вершины многогранника решений) системы ограничений задачи линейного программирования к другому базисному решению до тех пор, пока целевая функция не примет оптимального значения (максимума или минимума). Он используется как элемент классического решения СЦЛП на начальном этапе и далее после каждого нового ограничения на целочисленность. Симплекс-метод является универсальным методом, которым можно решить любую задачу линейного программирования [4].

Будем считать, что любое неотрицательное решение системы ограничений называется допустимым решением. Пусть имеется система m ограничений с n переменными ($m < n$). Допустимым базисным решением является решение, содержащее m неотрицательных основных (базисных) переменных и $n - m$ неосновных (небазисных, или свободных) переменных. Неосновные переменные в базисном решении равны нулю, основные же переменные, как

правило, отличны от нуля, то есть являются положительными числами. Любые m переменных системы m линейных уравнений с n переменными называются основными, если определитель из коэффициентов при них отличен от нуля. Тогда остальные $n - m$ переменных называются неосновными (или свободными).

Алгоритм решения симплекс-метода:

1. Привести задачу линейного программирования к канонической форме. Для этого перенести свободные члены в правые части (если среди этих свободных членов окажутся отрицательные, то соответствующее уравнение или неравенство умножить на -1) и в каждое ограничение ввести дополнительные переменные (со знаком «плюс», если в исходном неравенстве знак «меньше или равно», и со знаком «минус», если «больше или равно»).
2. Если в полученной системе m уравнений, то m переменных принять за основные, выразить основные переменные через неосновные и найти соответствующее базисное решение. Если найденное базисное решение окажется допустимым, перейти к допустимому базисному решению.
3. Выразить функцию цели через неосновные переменные допустимого базисного решения. Если отыскивается максимум (минимум) линейной формы и в её выражении нет неосновных переменных с отрицательными (положительными) коэффициентами, то критерий оптимальности выполнен и полученное базисное решение является оптимальным – решение окончено. Если при нахождении максимума (минимума) линейной формы в её выражении имеются одна или несколько неосновных переменных с отрицательными (положительными) коэффициентами, перейти к новому базисному решению.
4. Из неосновных переменных, входящих в линейную форму с отрицательными (положительными) коэффициентами, выбирают ту, которой соответствует наибольший (по модулю) коэффициент, и переводят её в основные. Переход к шагу 2.

В своём чистом виде симплекс-метод является весьма неэффективным для вычислений, в связи с чем были разработаны всяческие модификации алгоритма, самыми известными из которых являются модифицированный симплекс-метод (Revised Simplex Method) [4] и двойственный симплекс-метод (Dual Simplex Method) [4].

Метод внутренней точки — это метод, позволяющий решать задачи выпуклой оптимизации с условиями, заданными в виде неравенств, сводя исходную задачу к задаче выпуклой оптимизации [4]. В СЦЛП он выступает как альтернатива симплекс-методу в рамках внутреннего шага решения задачи линейного программирования.

Согласно методам внутренней точки (иначе называемым методами барьерных функций), исходную для поиска точку можно выбирать только внутри допустимой области.

Выбор начальной точки поиска осуществляется в зависимости от формулировки задачи. При отсутствии ограничений или их преобразовании к функциям штрафа с внешней точкой начальная точка выбирается произвольно. При наличии ограничений или их преобразовании к функциям штрафа с внутренней точкой начальная точка выбирается внутри допустимой области.

При этом множество точек делится на допустимые и недопустимые в зависимости от ограничений. В свою очередь множество допустимых точек в зависимости от ограничений также делится на граничные и внутренние.

Метод внутренней точки не ограничен только линейным программированием в отличие от симплекс-метода.

В последнее время растет интерес к методам оптимизации, основанным на ***машинном обучении*** (Machine Learning – ML). В широком смысле их можно разделить на методы внутри решателей, методы вне решателей и методы, заменяющие решатели.

Решатели смешанного линейно-целочисленного программирования (Mixed-Integer Linear Programming – MILP) зависят от параметров конфигурации для управления их внутренними алгоритмами. Решения и связанные с ними затраты или время выполнения существенно зависят от параметров конфигурации, даже если задача имеет одинаковое количество переменных решения и ограничений.

Мотивацией применения машинного обучения является обнаружение повторяющихся закономерностей и характеристик в задачах, которые решаются неоднократно [6]. Исследователи смогли добиться многообещающих результатов либо путем интеграции моделей в цикле ветвей и границ решателя, либо путем замены решателя сквозным алгоритмом, который принимает необработанный экземпляр задачи в качестве входных данных и напрямую или итеративно выводит допустимое решение.

Также методы машинного обучения используются для улучшения работы других ML-методов. Например, разные подходы обучения с подкреплением могут быть использованы для улучшения работы метода ветвей границ, заменяя методы и правила выбора переменных для ветвления, узлов для обработки и др. проблем метода ветвей и грани [2].

Метод имитации отжига относится к метаэвристическим подходам решения оптимизационных задач. Идея алгоритма имитации отжига заимствована из исследований поведения атомов металла в процессе его отжига. В металле, нагретом до температуры, превышающей точку его плавления, атомы находятся в беспорядочном движении [7]. При этом, как и во всех физических системах, они стремятся к состоянию минимума энергии (единому кристаллу), но при высоких температурах энергия атомных движений препятствует этому. В процессе постепенного охлаждения металла возникают все более низкоэнергетические состояния, пока не будет достигнуто низшее из возможных состояний, глобальный минимум.

Метаэвристические методы, к которым относится ***генетический алгоритм***, в своей основе используют итерационную технику улучшения результатов [8]. В течение одной итерации они ищут решение, лучшее в окрестностях данного. Если такое решение найдено, оно становится текущим и начинается новая итерация. Это продолжается до тех пор, пока прирост целевой функции не уменьшится практически до нуля или не выполнится заданное количество итераций. Очевидно, что такие методы ориентированы на поиск только локальных оптимумов, причём положение найденного оптимума зависит от стартовой точки. Глобальный же оптимум может быть найден только случайно. Для повышения вероятности нахождения глобального оптимума используется множественный эксперимент с различными начальными точками, что существенно увеличивает время поиска.

Свободные решатели на языке Python

Рассмотрим классификацию оптимизационных задач применительно к библиотекам, поддерживающим язык Python (см. рис.1):

1. Безусловная оптимизация,
2. LP (Linear Programming) – линейное программирование,
3. MILP (Mixed-Integer Linear Programming) — смешанное целочисленное линейное программирование, это задача LP, в которой часть переменных являются целочисленными,
4. NLP (Nonlinear Programming) — нелинейное программирование, возникает, когда хотя бы одна из функций нелинейна,
5. MINLP (Mixed-Integer Nonlinear Programming) — смешанное целочисленное нелинейное программирование, возникает, как и в случае с MILP, когда часть переменных принимает целочисленные значения.

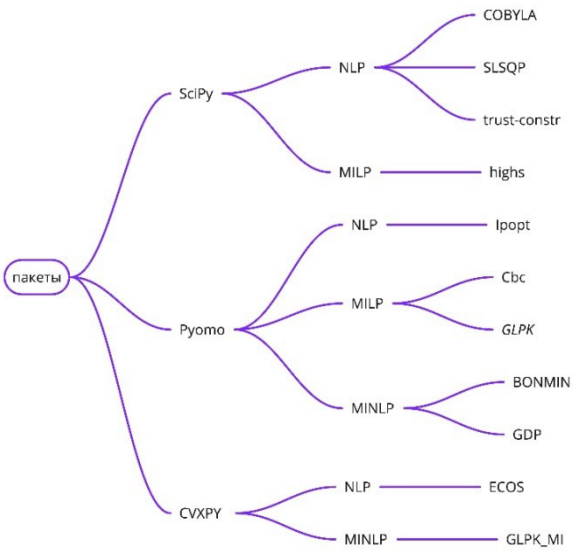

Рис.1. Классификация методов и их реализация в пакетах Python

SciPy — одна из первых библиотек в Python, с знакомства с которой начинается работа у специалистов в области Data Science: она содержит большой набор функций для научных вычислений, в том числе имеет инструменты для решения оптимизационных задач, находящихся в модуле scipy.optimize. SciPy – общий числовой пакет для Python с некоторой поддержкой оптимизации. SciPy содержит модули для оптимизации, линейной алгебры, интегрирования, интерполяции, специальные функции, быстрое преобразование Фурье, сигнал и обработку изображений, решатели ODE и другие задачи, распространенные в науке и технике.

SciPy основан на объекте массива NumPy и является частью стека NumPy, который включает такие инструменты, как Matplotlib, pandas и SymPy и расширяющийся набор научных вычислительных библиотек. В этом модуле имеются методы для решения задач как нелинейного программирования (NLP), так и линейного программирования (LP), в том числе задач смешанного целочисленного линейного программирования (MILP).

Однако главная проблема использования SciPy – это необходимость приведения задачи к строгой канонической форме и загрузки ограничений и целевой функции как массивов NumPy. Кроме того, он не поддерживает параллельный режим работы и использование графических процессоров для оптимизации.

Среди решателей, которые поддерживают решение задач нелинейной оптимизации (NLP), можно выделить COBYLA, SLSQP, trust-constr. Все три метода реализованы в пакете scipy.optimize.minimize и определяются параметром method ('cobyla', 'slsqp' и 'trust-constr' соответственно).

COBYLA — это метод, позволяющий производить оптимизацию функции, градиент которой неизвестен, т.е., по сути, заниматься оптимизацией «черного ящика». Для практического применения важно, что этот метод:

- не поддерживает ограничения типа равенства, что связано с особенностями реализации решателя, который внутри себя вызывает SciPy. Эту проблему можно обойти, заменив ограничение на два ограничения вида неравенства, которое равносильно равенству,
- не поддерживает границы для переменных, задаваемых через параметр bounds в функции minimize (из модуля scipy.optimize) — их необходимо задавать через линейные ограничения, например с помощью LinearConstraint.

SLSQP и trust-constr — это методы уже второго порядка, соответственно, для них требуется существование и непрерывность 1-й и 2-й производных.

Pyomo — пакет, который содержит ряд инструментов для формулирования, решения и анализа оптимизационных моделей. Главная особенность – это удобный интерфейс для структурированного формирования оптимизационной задачи и поддержка большого количества решателей, в том числе коммерческих. Pyomo внутри себя преобразует сформулированную модель в формат, понятный для запускаемого решателя.

Pyomo входит в проект COIN-OR, содержащий ряд решателей, среди которых выделим два:

- Ipopt – находит локальные оптимумы в задаче NLP с помощью прямо-двойственного метода внутренней точки,
- CBC – решает задачи MILP на базе алгоритма, сочетающего в себе метод ветвей и границ и секущих плоскостей.

Также для решения задач LP/MILP имеется поддержка пакета GLPK. Отметим ещё один решатель bonmin, который построен на базе решателей CBC и Ipopt, что позволяет решать задачи MINLP.

Процесс построения оптимизационной модели в Pyomo состоит из следующих основных этапов:

- создание объекта оптимизационной модели,
- объявление переменных в этой модели с границами,
- формулирование целевой функции,
- описание ограничений,
- запуск решателя, решающего задачу.

CVXPY – данный пакет был реализован для решения задач выпуклой оптимизации. Отметим ещё один решатель bonmin, который построен на базе решателей Cbc и Ipopt, что позволяет решать задачи MINLP.

Для решения задачи, по аналогии с Pyomo, необходимо выполнить несколько шагов: определить переменные, задать целевую функцию и ограничения для формирования объекта оптимизационной задачи. После того как задача сформулирована, перед запуском решателя проверяется выпуклость целевой функции и ограничений с помощью правил DCP (Disciplined Convex Programming). По сути, это набор правил, которые однозначно гарантируют выпуклость функции, здесь стоит отметить, что эти правила – необходимые условия, но не достаточные, почему это так – рассмотрим в примере ниже. После проверки задача преобразуется для передачи решателю в формат, поддерживающий решение задач выпуклой оптимизации, в том числе CVXPY поддерживает коммерческие решатели. Полный список решателей можно посмотреть на этой странице, там же указаны все типы задач, которые позволяет решать CVXPY. Обратим внимание, что для данного пакета не так важно наличие непрерывности первой и второй производной, важна только выпуклость.

Если в задаче NLP первые и вторые производные являются гладкими, то здесь хорошо себя зарекомендовал себя Ipopt в пакете Pyomo. Если в NLP функции не гладкие, но выпуклые, то для решения задачи подойдет CVXPY. Scipy хорошо справляется с небольшими задачами, а COBYLA можно использовать для задач без градиента целевой функции. Для задачи MILP конкурирующими являются решатели GLPK, Cbc, HiGHS – здесь заранее сказать, какой из них будет более производительным для нашей задачи непросто, надо пробовать, перебирать. SciPy с точки зрения интерфейса не

очень удобен, если имеются сложные ограничения. Результат качественного сравнения решателей приведен в таблице 1.

Таблица 1

Сравнение решателей

Пакеты в Python	Решатель (метод)	NLP	LP	MILP	MINLP
SciPy	COBYLA	+	-	-	-
SciPy	SLSQP	+	-	-	-
SciPy	trust-constr	+	-	-	-
SciPy	HiGHS	+	+	+	-
Pyomo	Ipopt	+	+	-	-
Pyomo, CVXPY	GLPK	-	+	+	-
Pyomo, CVXPY	CBC	-	+	+	-
CVXPY	Ecos	+-	+	+	-
Pyomo	Bonmin	+	+	+	+

Также весьма популярным бесплатным решателем является PuLP. Он написан на Python, может читать и генерировать MPS- и LP-файлы и вызывать GLPK, COIN-OR CLP/CBC, CPLEX, GUROBI, MOSEK, XPRESS, CHOCO, MIPCL, HiGHS, SCIP/FSCIP для решения задач линейного программирования.

Python-MIP является другим распространённым пакетом для Python, вдохновлённым синтаксисом PuLP (что делает весьма простой миграцию моделей из PuLP в MIP), но при этом предоставляющим некоторые продвинутые функции, такие как генерации сечений, ленивые ограничения и др. При установке пакета также устанавливается и CBC решатель, но помимо него можно подключить Gurobi при наличии лицензии.

Т.к. оба эти пакета содержат решатели, не являющиеся эксклюзивными для них, в таблице они не приведены.

Google OR-Tools – это популярный программный пакет с открытым исходным кодом для оптимизации, написанный на C++, но поддерживающий языки программирования Python, C# и Java. Он предназначен для решения сложных задач маршрутизации транспортных средств, потоков, целочисленного и линейного программирования, а также программирования в ограничениях.

Смоделировав проблему на выбранном языке программирования, можно использовать для ее решения любой из полудюжины решателей: коммерческие

решатели, такие как Gurobi или CPLEX, или решатели с открытым исходным кодом, такие как SCIP, GLPK, CP-SAT или GLOP (Google Linear Optimization Package).

Коммерческие пакеты программного обеспечения MILP

Самые ранние реализации алгоритмов ветвей и границ коммерческого уровня для решения MILP были выполнены Beale&Small [9]. С ранними пакетами программного обеспечения MILP, такими как UMPIRE, SCICONIC и MPSX/370, можно ознакомиться в работе Форреста и Томлина [10]. В этих пакетах программного обеспечения MILP реализовано множество эффективных методов ветвления и выбора узлов, некоторые из которых используются до сих пор. Большой шаг вперед в технологии развития решений MILP представлен в работах [11] и [12]. В этих статьях были освещены многие аспекты предварительной обработки. Еще одна важная с вычислительной точки зрения работа была проведена Баласом и др. [13], демонстрируя, что неравенства Гомори [14] можно эффективно использовать для отсечения плоскости. Современные коды MILP синтезировали многие из этих подходов, взяв лучшие и воплотив их в пакеты программного обеспечения коммерческого уровня [15].

Важным аспектом разработки программного обеспечения для решения MILP является пользовательский интерфейс. Диапазон целей программного обеспечения MILP достаточно широк, поэтому количество типов пользовательского интерфейса также велико.

Как правило, пользователи реализуют задачи MILP, используя решатель в качестве «черного ящика». Можно вызвать программное обеспечение из стороннего пакета или встроить решатель в собственные приложения, чтобы программное обеспечение использовало вызываемую библиотеку. Также пользователь может вносить собственные правки в алгоритм для конкретной задачи.

Ниже представлен краткий обзор наиболее широко используемого коммерческого программного обеспечения MILP.

Общая алгебраическая система моделирования (GAMS) представляет собой систему моделирования высокого уровня для математической оптимизации. GAMS предназначен для моделирования и решения задач линейной, нелинейной оптимизации и оптимизации со смешанными целыми числами. Система предназначена для сложных крупномасштабных приложений моделирования и позволяет пользователю создавать большие устойчивые модели, которые могут быть адаптированы к новым ситуациям. Система доступна для использования на различных компьютерных платформах. Модели могут портироваться с одной платформы на другую. GAMS был первым языком алгебраического моделирования (AML) и формально аналогичен широко используемым языкам программирования четвертого поколения.

GAMS содержит интегрированную среду разработки (IDE) и подключен к группе сторонних решателей для оптимизации. Среди этих решателей – BARON, COIN-OR, CONOPT, COPT Cardinal Optimizer, CPLEX, DICOPT, MOSEK, SNOPT, SULUM и XPRESS. AMS – это высокоуровневая система моделирования для математического программирования и оптимизации. Она состоит из языкового компилятора и ряда связанных с ним решателей. Язык моделирования GAMS позволяет разработчикам моделей быстро переводить задачи оптимизации реального мира в компьютерный код. Затем компилятор языка GAMS переводит этот код в формат, понятный решателям. Такая архитектура обеспечивает большую гибкость, позволяя изменять используемые решатели без изменения формулировки модели. Стоимость GAMS: основной модуль 3500$, дополнительный от 3500$ до 14000$.

CPLEX – это коммерческий пакет программного обеспечения MILP, принадлежащий и распространяемый компанией IBM. Реализует интерфейсы C, C++, Java, .NET, Matlab, Python, Microsoft Excel. Основной алгоритм основан на методе ветвления и границ, он содержит полный набор предварительных решений, также метод секущей плоскости и эвристические методы для поиска возможных решений. Специальный алгоритм поиска в CPLEX, называемый динамическим, содержит средства автоматической настройки параметра алгоритма MILP для определенного класса. CPLEX может распараллелить решение по методу ветвей и границ с использованием нескольких потоков в детерминированном или недетерминированном режиме. Последней примечательной особенностью CPLEX является его способность перечислять несколько решений, близких к оптимальному, и сохранять их. Следует отметить, что стоимость данного коммерческого продукта на сайте производителя не указана и предоставляется по запросу. У дилеров указана цена 12,910.28 $. Подписка за внутреннее приложение на одного человека составляет 269 $. Лицензии по подписке предоставляются со скидкой 50% от обычной цены. Срок действия этих лицензий автоматически истекает через 12 месяцев.

Gurobi Optimizer содержит относительно новый решатель MILP, созданный с нуля, для использования современной технологии многоядерной обработки. Поддерживает языки программирования C, C++, Java, Python, .NET, Matlab. Gurobi содержит все расширенные алгоритмические функции, включая сечения плоскости, эвристики и методы поиска. Gurobi может выполнить вычисления по методу ветвей и границ, используя несколько вычислительных потоков. Параллельный поиск осуществляется детерминированным образом. Интерактивная оболочка Gurobi реализована как набор расширений оболочки Python. Gurobi также доступен «по требованию» с использованием Amazon Elastic Compute Cloud. Стоимость коммерческого продукта по запросу, первоначально пробная версия на 30 дней бесплатно. Крупные компании отзываются о нем как о самом быстром оптимизаторе для высокоразмерных задач.

Компания LINDO Systems предлагает решатель MILP как часть своего LINDO API. Он поддерживает языки C, VisualBasic, Matlab. Решатель MILP предлагает пятнадцать различных типов секущих плоскостей, а также различные методы предварительной обработки и правила ветвления. Стоимость коммерческого продукта составляет от 495 $ до 4995,5 $ в зависимости от комплектации и целей использования.

MOSEK ApS – компания, специализирующаяся на общей программном обеспечении для математической оптимизации, а в их пакет программного обеспечения входит решатель для MILP. Он поддерживает языки C, C++, Java, .NET, Python.

Решатель включает шесть правил выбора узлов, одиннадцать типов секущих плоскостей и другие эвристики, а также передовые методологии ветвления, например, сильное ветвление. Серверная лицензия бессрочна, и ее стоимость составляет 7400 $ или 8200 $ в зависимости от пакета. Она является бессрочной, но не включает обновления. Обслуживание в год требует 1850 $ – 2050 $ и является необязательным, но позволяет обновлять решатель до новых версий, а также дает право на неограниченную смену хостинга и предоставляет другие преимущества. Плавающая лицензия привязана к определенному компьютеру, действующему как сервер токенов лицензий. Ее стоимость 1850 $ – 2050 $, а обслуживание в год 462,50 $ – 512,50 $.

Ниже стоимостные характеристики сведены в единую таблицу 2.

Таблица 2

Стоимость решателей

Название	Компания	Стоимость
GAMS	GAMS SoftwareGmbH	$3 500 – базовая оболочка + до $14 000 за отдельные решатели
CPLEX	IBM	$12 910.28 + $269 за человека
Gurobi Optimizer	Gurobi Optimization, LLC	$8 395.00
LINDO	LINDOSystems	$4,995 + $1,500 за отдельные методы
Mosek	MosekApS	$8200 + $2050 поддержка в год

В основном все компании не разглашают цену своих продуктов, однако ее можно увидеть у некоторых дилеров. Как правило, ценовая политика строится вокруг базовой цены с возможностью докупать дополнительные модули других компаний или интегрировать открытые решатели.

Сравнение решателей на бенчмарках

В ответ на потребности исследователей в доступе к реальным смешанным целочисленным программам Роберт Э. Биксби, Е.А. Бойд и Р.Р. Индовина

создали в 1992 году MIPLIB, доступную в электронном виде библиотеку как чистых, так и смешанных целочисленных задач [16]. С момента своего появления MIPLIB стал стандартным набором тестов, используемым для сравнения производительности оптимизаторов СЦЛП. Его доступность стала важным стимулом для исследователей в этой очень активной области. В настоящее время библиотека выпущена в шестом издании в результате совместных усилий Университета штата Аризона, COIN-OR, CPLEX, FICO, Gurobi, MathWorks, MIPCL, MOSEK, NUOPT, SAS и Института Цузе в Берлине.

По практическим соображениям эталонные 10 примеров были выбраны с учетом различных ограничений, касающихся разрешимости, численной стабильности и др.

Далее приведен перечень 10 эталонных задач из бенчмарков для последующего тестирования:

1) Задача разделения рынка markshare_4_0 (https://miplib.zib.de/instance_details_markshare_4_0.html),
2) pk1 (https://miplib.zib.de/instance_details_pk1.html),
3) Задача с расписанием общественного транспорта timtab1 (https://miplib.zib.de/instance_details_timtab1.html),
4) Модель для решения примера комбинаторной игры EnLightenlight_hard (https://miplib.zib.de/instance_details_enlight_hard.html),
5) mas76 (https://miplib.zib.de/instance_details_mas76.html),
6) neos5 (https://miplib.zib.de/instance_details_neos5.html),
7) sp150x300d (https://miplib.zib.de/instance_details_sp150x300d.html),
8) neos-911970 (https://miplib.zib.de/instance_details_neos-911970.html),
9) tr12-30 (https://miplib.zib.de/instance_details_tr12-30.html),
10) exp-1-500-5-5 (https://miplib.zib.de/instance_details_exp-1-500-5-5.html).

Не все эти задачи имеют предметное описание решаемых задач, но для целей верификации и сравнительного сопоставления с другими решателями этого и не требуется.

В таблице 3 представлены результаты выполнения 10 бенчмарков. Сравнение проводилось на ЭВМ со следующим аппаратным обеспечением: 12th GenIntel (R) Core (TM) i5-1240P 1.70 GHz, 16,0 GB (15,7 GB usable), диск 500 Гб SSD.

Таблица 3

Сравнение точности и скорости работы решателей

Название оптимизационной задачи	Значение целевой функции в Benchmark	Значение целевой функции SciPyHIGHS	Значение целевой функции OR-ToolsSCIP	Значение целевой функции Python-MIPCBC	Время работы SciPy HIGHS	Время работы OR-Tools SCIP	Время работы Python-MIPCBC
sp150x300d	69,00	69,00	69,00	69,00	0,06	1,14	900,00
markshare_4_0	1,00	1,00	1,00	1,00	108,84	96,07	17,26
pk1	11,00	11,00	11,00	11,00	203,75	89,17	51,83
timtab1	764 772,00	764 774,00	764 772,00	788305,99	54,16	140,55	900,00
enlight_hard	37,00	37,00	37,00	37,00	11,22	0,00	252,82
mas76	40 005,05	40 005,05	40 005,05	40005,05	133,00	100,97	37,08
neos5	15,00	15,00	15,00	15,00	138,66	212,22	389,02
neos-911970	54,76	54,76	54,76	55,26	11,27	301,44	900,00
tr12-30	130 596,00	130 599,00	130 596,00	133788,00	353,97	710,07	900,00
exp-1-500-5-5	65 887,00	65 887,00	65 887,00	66472,00	6,73	2,46	900,00

Анализ результатов показал, что нет одного доминирующего решателя. Разница в качестве и скорости работы обусловлена тем, что используются разные, в том числе рандомизированные, эвристики для ветвлений и отсечений. Поэтому разработчику прикладных программных решений целесообразно обеспечить возможность интеграции широкой палитры разнообразных решателей и эвристик, и в прикладной задаче необходимо подбирать метод и настраивать его гиперпараметры.

Также стоит отметить, что часто открытые пакеты для Python используют одни и те же решатели, для которых просто создаётся обёртка для языка Python. Например, решатель GLPK доступен и в Pyomo, и в OR-Tools, HIGHS доступен в SciPy и PuLP, а CBC – в Pyomo, PuLP и Python-MIP. Из-за этого часто на выбор пакета может влиять интерфейс или синтаксис, используемый пакетом, или некоторый дополнительный функционал.

Так, например, SciPy работает исключительно с матрицами и векторами (а точнее массивами NumPy или списками Python), в то время как такие пакеты, как Pyomo или Python-MIP, предоставляют возможность создания моделей в более удобной для чтения пользователя форме; позволяют задавать названия переменных и др. подобные вещи. С другой стороны, некоторые пакеты обладают весьма удобным дополнительным функционалом, как, например,

чтение MPSфайлов – функционал, реализованный в OR-Tools, PuLP или Python-MIP, но отсутствующий в Pyomo и SciPy.

Заключение

Были рассмотрены основные методы решения СЦЛП и программные пакеты для решения данного типа задач.

Наличие большого количества программных продуктов на этом рынке говорит о его перспективности. Продукты предоставляют оболочки для интеграции с другими решателями и с возможностью вызова из наиболее массовых языков программирования.

Основная проблема метода ветвей и границ и его аналогов – это сложность создания их эффективной параллельной версии, а также сложность выбора правил и эвристик для его эффективной работы.

Метаэвристические методы имеют тот же недостаток и, кроме того, не гарантируют нахождение точного решения. О сложностях создания универсальных метаэвристик говорит и тот факт, что они слабо представлены на рынке популярных универсальных коммерческих программных продуктов.

Перспективным является создание методов оптимизации методов на основе машинного обучения, так как они допускают эффективную параллельную реализацию.

Библиографический список

1. Мельничук А.В., Сивакова Т.В., Судаков В.А. Решение задач оптимизации с использованием мультиагентных моделей // Препринты ИПМ им.М.В.Келдыша. – 2019. – № 100. – 16 с. – doi:10.20948/prepr-2019-100.
2. Huang L., Chen X., Huo W., Wang J., Zhang F., Bai B. and Shi L. Branch and bound in mixed integer linear programming problems: a survey of techniques and trends // Preprint submitted to Discrete Optimization. – 2021. – URL: https://arxiv.org/abs/2111.06257.pdf (дата обращения 15.03.2024).
3. Хоролич Г.Б. Подходы к решению задач смешанного целочисленного линейного программирования // Развитие современной науки: теоретические и прикладные аспекты: сборник статей студентов, магистрантов, аспирантов, молодых ученых и преподавателей. / Под общей редакцией Т.М. Сигитова. Выпуск 14. – Пермь: ИП Сигитов Т.М., 2017. – С. 5-7.
4. Achterberg T. Constraint Integer Programing. Technische Universität Berlin. – 2007. – URL: https://doi.org/10.14279/depositonce-1634 (дата обращения 12.03.2024).
5. Bertsimas D., Tsitsiklis J.N. Introduction to Linear Optimization. – Athena Scientific, 1997. – 608 p.

6. Белозеров И.А., Судаков В.А. Машинное обучение с подкреплением для решения задач математического программирования // Препринты ИПМ им.М.В.Келдыша. – 2022. – № 36. – 14 с. – doi:10.20948/prepr-2022-36.
7. Рыбак Л.А., Ержуков В.В.,Чичварин А.В Эффективные методы решения задач кинематики и динамики робота-станка параллельной структуры. – М.: Физматлит, 2011. – 148 с.
8. Цветкова Н.В. Методика исследования разработки технологических карт на базе линейного программирования // Вестник научных конференций. – 2019. – № 10-4(50). – С. 129-130. – URL:https://ukonf.com/doc/cn.2019.10.04.pdf (дата обращения 15.03.2024).
9. Beale E.M.L., Small R.E. Mixed integer programming by a branch and bound method // Proceedings IFIP Congress 65, Kalenich W.H. (editor). 1965. – vol. 2. – p. 450–451.
10. Forrest J.J.H., Tomlin J.A. Branch and bound, integer and non-integer programming. Annals of Operations Research. 2007. – vol. 149. – p.81–87. – doi:10.1007/s10479-006-0112-x.
11. Crowder H., Johnson E., Padberg M.W. Solving large scale zero-one linear programming problem // Operations Research. – 1983. – vol. 31. – p.803–834.
12. Van Roy T.J., Wolsey L.A. Solving mixed integer programming problems using automatic reformulation // Operations Research. – 1987. – vol. 35. – p. 45– 57.
13. Balas E., Ceria S., Cornu´ejols G., Natraj N. Gomory cuts revisited. Operations Research Letters. – 1996. – vol. 19. –p. 1–9.
14. Gomory R.E. An algorithm for the mixed integer problem. Technical Report RM-2597. – The Rand Corporation, Santa Monica, CA, 1960.
15. Bixby R., Rothberg E. Progress in computational mixed integer programming – a look back from the other side of the tipping point // Annals of Operations Research. – 2007. – vol. 149. – p. 37–41. – doi:10.1007/s10479-006-0091-y.
16. MIPLIB 2017 – The Mixed Integer Programming Library. URL: https://miplib.zib.de/ (дата обращения 20.03.2024).

Оглавление