ТЕМА 7. ДИСКРЕТНАЯ ОПТИМИЗАЦИЯ

Одна из наиболее актуальных проблем математического моделирования — решение оптимизационных задач, заданных на дискретных множествах (дискретная оптимизация). Алгоритмы решения таких задач связаны с отбором или упорядочением элементов различных дискретных множеств. В связи с этим задачи дискретной оптимизации и алгоритмы их решения называют комбинаторными.

Для комбинаторных задач характерна расчлененность области определения на отдельные элементы и (или) конечность множества значений функции. Как правило, алгоритмы их решений не имеют ядра, т.е. некоторого набора основных положений, теорем, правил, позволяющих получить общие результаты и рекомендации. Единственным общим методом решения комбинаторных задач является полный перебор вариантов. Однако количество требующих рассмотрения альтернатив возрастает с увеличением размерности задачи столь быстро, что время реализации алгоритмов полного перебора даже на самых быстродействующих ЭВМ слишком велико для практической реализации (так называемое проклятие размерности).

Вместе с тем получены эффективные принципы организации вычислений и решающие правила, отражающие особенности того или иного класса задач, что позволяет резко сократить (в среднем!) объем вычислений и сделать возможным практическое решение такого рода задач. Некоторые из них рассматриваются в данном разделе.

Дискретное программирование — область математики, занимающаяся экстремальными задачами на конечных множествах. Дело в том, что на практике рассматривается все больше задач оптимизации типа

$$\mathbf{X}^* = \arg\min W(\mathbf{X}) \,|\, \mathbf{X} \in D,$$

в которых компоненты неизвестного вектора X (все или часть из них) являются элементами некоторого дискретного множества *Z*. Такие задачи возникают тогда, когда рассматриваются неделимые объекты и эта неделимость является принципиальной. Можно выделить два основных класса целочисленных задач: задачи с неделимостями и задачи выбора вариантов — вариантные задачи. Примерами первых являются, например, задачи выбора маршрута, раскроя, планирования выпуска изделий, в которых искомые переменные в принципе существуют только целыми. Задачи же выбора содержат целочисленные неизвестные, которые могут принимать лишь одно из двух значений «1» или «О». Примеры таких задач: выбор типа оборудования, выбор варианта действий и т.п.

Последствия требования целочисленности параметров и переменных в целом столь значительны, что часто приводят к необходимости кардинального пересмотра методов решения, а иногда и самой методологии поиска оптимального решения.

Целочисленное линейное программирование

Постановка задачи целочисленного линейного программирования. В целом ряде случаев в задаче линейного программирования присутствует

дополнительное условие о том, что все или некоторые переменные должны быть целочисленными. Эти условия возникают тогда, когда дробные значения переменных противоречат их физическому смыслу. С точки зрения математики — это условие приводит к принципиально новому классу моделей — целочисленному программированию. Принципиальная особенность таких задач заключается в том, что *X* принадлежит дискретному множеству, а это в свою очередь требует пересмотра многих положений теории линейного программирования.

В общем случае задача целочисленного линейного программирования (ЦЛП) имеет вид

$$\begin{aligned} &W(\mathbf{X}) = \mathbf{CX} \to \max, \\ &\mathbf{AX} = \mathbf{B}, \\ &\forall x_j \geq 0, \\ &\exists x_j - \text{целые}. \end{aligned} \qquad (1)$$

На практике последнее условие в (1) часто заменяется условием булевости переменных, т.е. x - принимает одно из двух возможных значений: 0 или 1.

Метод ветвей и границ. Среди применяемых сегодня методов решения задач ЦЛП наибольшее распространение получил метод ветвей и границ. Идея метода заключается в следующем. Пусть имеется пара задач: ЦЛП и соответствующая ее непрерывная задача ЛП, т.е.

$$\begin{aligned} &W(\mathbf{X}) = \mathbf{CX} \to \max, \\ &\mathbf{AX} = \mathbf{B}, \\ &\forall x_j \geq 0, \\ &\exists x_j - \text{целые}; \end{aligned} \qquad \begin{cases} L(\mathbf{X}) = \mathbf{CX} \to \max, \\ \mathbf{AX} = \mathbf{B}, \\ \forall x_j \geq 0. \end{cases}$$

Очевидно, что всегда выполняется условие L(X) > W(X). Таким образом, решение соответствующей задачи ЛП обеспечивает получение оценки решения ЦЛП сверху (дает верхнюю границу для ЦЛП). Область допустимых решений *R* разбивается на непересекающиеся подмножества *R*

Метод секущих плоскостей

Метод секущих плоскостей является точным методом решения смешанных задач целочисленного программирования, в которых матрица A не является вполне унимодулярной. Современные практические методы решения задач целочисленного программирования используют алгоритмы ветвлений и отсечений, которые объединяют метод секущих плоскостей с методом ветвей и границ, обсуждаемым в следующем разделе. Метод секущих плоскостей решает ослабленную задачу LP и добавляет линейные ограничения, которые приводят к оптимальному решению.

Метод секущих плоскостей начинается с решения x_c^* ослабленной задачи LP, которое должно быть вершиной, удовлетворяющей системе уравнений Ax = Ь. Если V компонентов вектора x являются целочисленными, то он также является оптимальным решением исходной смешанной задачи

целочисленного программирования, и все готово. Пока V компонентов вектора x не являются целыми, мы находим гиперплоскость, относительно которой вектор x лежит на одной стороне, а все возможные дискретные решения — на другой. Эта секущая плоскость является дополнительным линейным ограничением для исключения x_c^*. Затем расширенная задача LP решается для новой точки x_c^*.

На каждой итерации алгоритма вводятся секущие плоскости, которые делают нецелые компоненты вектора x_c^* недопустимыми при сохранении условий допустимости ближайших целочисленных решений и остальной части допустимого множества. Затем решается целочисленная задача, дополненная ограничениями, заданными секущими плоскостями, и находится новое ослабленное решение.

Динамическое программирование

Динамическое программирование — это метод, который можно применять к задачам с оптимальной подструктурой и перекрывающимися подзадачами. Задача имеет оптимальную подструктуру, если оптимальное решение может быть построено из оптимальных решений ее подзадач.

Термин динамическое программирование был выбран Веллманом для отражения изменяющегося во времени аспекта задач, к которым он его применял, и для того, чтобы избежать иногда негативных коннотаций в словах, таких как исследование и математика.

Задача с перекрывающимися подзадачами, решаемая рекурсивно, требует многократного решения одной и той же подзадачи. Вместо того чтобы экспоненциально много раз перебирать множество потенциальных решений, динамическое программирование либо сохраняет решения подзадач и, таким образом, избегает необходимости заново их решать, либо рекурсивно создает оптимальное решение за один проход. Задачи с рекуррентными отношениями часто имеют перекрывающиеся подзадачи. Пример показан на рис.1.

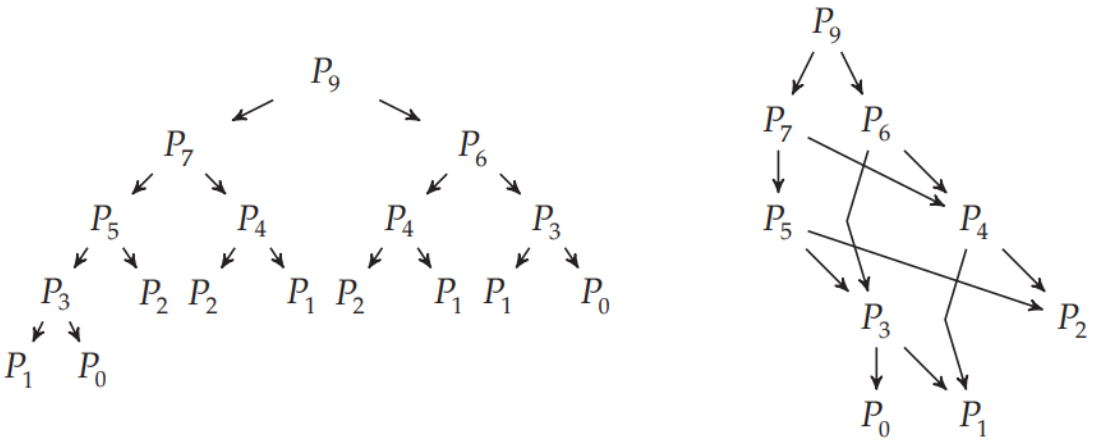

Рис.1. Вычислить *n*-й член *последовательности Падована*, $P_n = P_{n-2} + P_{n-3}$, с $P_0 = P_1 = P_2 = 1$, можно, вычислив все подвыражения (*слева*). Более эффективный подход состоит в том, чтобы вычислять подвыражения один раз и повторно использовать их значения в последующих вычислениях, используя перекрывающуюся подструктуру задачи (*справа*)

Динамическое программирование может быть реализовано как сверху вниз, так и снизу-вверх. Нисходящий подход начинается с желаемой задачи и рекурсивно сводится ко все меньшим и меньшим подзадачам. Решения подзадач хранятся таким образом, что, когда нам дается новая подзадача, мы можем либо получить вычисленное решение, либо решить и сохранить его для будущего использования. Восходящий подход начинается с решения меньших подзадач и использует их решения для получения решений больших задач.