К РЕШЕНИЮ ЗАДАЧ НА ГРАФАХ С ПОЗИЦИЙ КЛАССИЧЕСКОГО ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

В. М. Ракецкий
Кафедра информатики и прикладной математики, Брестский государственный технический университет
Брест, Республика Беларусь
E-mail: rvm@bstu.by

На примере транспортной задачи в сетевой постановке демонстрируется подход к решению экстремальных задач на графах с классических позиций линейного программирования. Предлагаемый подход может использоваться, в принципе, для любых задач на графах и сетях, в которых возникают системы линейных уравнений и неравенств, отражающие структуру графа.

Введение

Общепринятой является точка зрения, что решение задач на графах с классических позиций (посредством сведения к общим задачам математического, в частности, линейного программирования) является неэффективным. Поэтому для решения подобных задач, как правило, разрабатываются специальные методы, учитывающие их графовую природу.

Разработка таких методов и особенно их реализация в виде компьютерных программ требует совершенно иной техники, нежели разработка и программная реализация методов, основанных на классическом подходе. В силу этого методы и алгоритмы, разработанные для классических задач, требуют специальной адаптации для задач на графах. Вместе с тем учет структуры ограничений, возникающих при переходе от экстремальной задачи на графе к задаче математического программирования, позволяет избежать проблем, связанных с неэффективным использованием памяти компьютера, большим объемом вычислений и накоплением вычислительных погрешностей при применении классических методов для решения задач на графах. Впервые это было показано при исследовании квадратичной задачи на графе [1,2], возникающей в геодезической практике. Для решения этой задачи был предложен экономичный, численно устойчивый алгоритм, не требующий навыков работы с графами, основанный на прямом методе решения общей задачи квадратичного программирования [3].

Позже появилось понимание, что техника, использованная при разработке алгоритма решения задачи [1,2], может применяться при решении других оптимизационных задач на графах. Ниже демонстрируется её применение для решения транспортной задачи в сетевой постановке с одним источником и одним стоком.

I. Транспортная задача в сетевой форме и ее матричный аналог

Пусть $S = \{X, U\}$ – транспортная сеть. Здесь $X = \{1, 2, \ldots, n\}$ – множество узлов, узел 1 – источник с интенсивностью поставки a, узел n – сток с интенсивностью потребления a, узлы $2, 3, \ldots, n-1$ – транзитные; $U = \{u_1, u_2, \ldots, u_m\}$ – множество дуг, при этом каждая дуга $u \in U$ обладает тремя характеристиками: пропускной способностью $d(u)$, транспортными издержками $c(u)$ за единицу перемещаемого по дуге груза и дуговым потоком $x(u)$ – количеством перемещаемого по дуге груза. Эквивалентом поставленной задачи в линейном программировании является задача

$$\begin{gathered} L(x) = \sum_{u \in U} c(u)x(u) \to min, \\ -\sum_{u \in U^-(1)} x(u) = -a; \\ \sum_{u \in U^+(k)} x(u) - \sum_{u \in U^-(k)} x(u) = 0, k = \overline{2, n-1}, \\ 0 \le x(u) \le d(u), u \in U. \end{gathered} \tag{1}$$

Здесь $U^+(k)$ – множество дуг, входящих в узел k, $U^-(k)$ – множество дуг, исходящих из узла k. Задача (1) является частным случаем общей задачи линейного программирования:

$$L(x) = c^T x \to min,\ Ax = b,\ 0 \le x \le d. \tag{2}$$

Методы решения задачи (2), основанные на классическом симплекс-методе, представляют собой последовательность итераций, на каждой из которых решаются системы линейных алгебраических уравнений вида:

$$A_б y = p, \tag{3}$$

$$A_б^T y = q, \tag{4}$$

где $A_б$ – базисная матрица системы основных ограничений задачи (2), y – вектор решения, p, q – правые части систем. Трудоемкость симплекс-метода и его модификаций напрямую зависит от того, насколько эффективно решаются системы (3), (4). При решении общих задач линейного программирования для того, что-

бы не решать «с нуля» системы (3), (4) используют различные приемы: хранят и пересчитывают обратную матрицу $A_б^{-1}$ (в различных формах), хранят и пересчитывают различные разложения базисной матрицы (в частности, LU-разложение). При решении специальных задач, когда это позволяет структура матрицы A, системы (3), (4) можно решать на каждой итерации заново.

II. Свойства матрицы системы ограничений

Взглянем подробнее на структуру матрицы A. Фактически это матрица инцидентности графа, из которой удалена строка, соответствующая стоку. Очевидны следующие свойства:

1. Матрица A состоит только из трех элементов: $0, 1, -1$.
2. В каждом столбце матрицы A не более двух ненулевых элементов. При этом дуге, которая не инцидентна стоку, соответствует столбец с двумя ненулевыми элементами (1 и -1), а дуге, которая инцидентна стоку, с одним ненулевым элементом (-1).
3. Отсутствуют одинаковые столбцы и строки.

Менее очевидными, но доказуемыми являются свойства:

4. $rank\, A = n - 1$ – строки матрицы A являются линейно независимыми.
5. Каждому дереву на сети S соответствует базисная матрица задачи (2) и, наоборот, каждой базисной матрице соответствует дерево на сети.

Ассоциация базисной матрицы с деревом сети позволяет сформулировать еще два свойства:

6. Базисная матрица обязательно содержит хотя бы одну строку, в которой находится только один ненулевой элемент.
7. Если удалить из базисной матрицы эту строку и столбец, соответствующий её ненулевому элементу, то в оставшейся матрице снова будет присутствовать по крайней мере одна строка с только одним ненулевым элементом.
8. Свойства 6), 7) верны и для матрицы $A_б^T$.

III. Алгоритм решения систем (3), (4)

Свойства 6) –8) позволяют предложить весьма простой алгоритм решения систем типа (3), (4). Опишем его для системы (3):

1. Подсчитаем и запомним для каждого уравнения системы (3) количество входящих в него (неизвестных) переменных.
2. Последовательно анализируем уравнения системы (3) на предмет количества входящих в них неизвестных переменных. Если для всех уравнений это количество равно 0, то система решена. Работа алгоритма окончена.
3. Для всех уравнений, в которых не известна только одна переменная
   (a) найдем значение неизвестной переменной;
   (b) уменьшим количество неизвестных переменных на 1 для всех уравнений системы, в которые входит найденная переменная.
4. Перейдем к п.2.

На каждой итерации этого алгоритма находится значение, по крайней мере, одной неизвестной переменной. Поэтому все решение будет найдено не более чем за $n - 1$ итерацию.

Заключение

Описанный в настоящей работе подход позволяет использовать для решения сетевой транспортной задачи любые методы линейного программирования, основанные на понятии базиса (опоры). Поскольку системы уравнений (3), (4) решаются без использования обратной базисной матрицы или какого-либо ее разложения, а элементы матрицы A задачи (2) легко моделируются по описанию графа задачи, соответствующие алгоритмы будут а) нетребовательными к памяти компьютера, б) численно устойчивыми, в) эффективными с вычислительной точки зрения: в силу свойства 1) матрицы A решение систем (3), (4) не требует операций умножения и деления.

1. Ракецкий, В. М. Прямой опорный метод решения сетевой задачи квадратичного программирования. / В. М. Ракецкий // Динамические системы: устойчивость, управление, оптимизация: тез. докладов Междунар. конференции. Минск, 29 сентября – 4 октября 2008 г. – Мн.: Институт Математики НАН Беларуси, 2008. – С. 139–140
2. Ракецкий, В. М. Алгоритм решения задачи квадратичного программирования на графе. / В. М. Ракецкий // Математические и физические методы исследований: научный и методический аспекты: сборник материалов межвузовской научно-практической конференции, посвященной 370-летию со дня рождения И. Ньютона; Брест, 22 марта 2013 г. Брест. гос. ун-т имени А. С. Пушкина; под общ. ред. Н. Н. Сендера. – Брест: БрГУ, 2013. – 226 с. С. 151–155.
3. Ракецкий, В. М. Прямой опорный метод квадратичного программирования. / В. М. Ракецкий // Проблемы оптимального управления: сб. научн. тр. – Минск: Наука и техника, 1981. – С. 318–335.