УДК 519.176

Основы решения задач оптимизации на графах в пакете компьютерной алгебры Wolfram Mathematica

Осипов Геннадий Сергеевич
Сахалинский государственный университет
д.т.н., заведующий кафедрой Информатики

Вашакидзе Нателла Семеновна
Сахалинский государственный университет
доцент кафедры Информатики

Филиппова Галина Викторовна
Сахалинский государственный университет
доцент кафедры Информатики

Аннотация
Изложена формальная постановка наиболее известных проблем оптимизации на графах – задача о кратчайшем пути, коммивояжера и о максимальном потоке. Практическая апробация методов решения задач выполнена в среде пакета *Wolfram Mathematica*, который является одной из наиболее мощных систем обработки информации методами символьной математики и компьютерной алгебры.
Ключевые слова: графы, алгоритмы оптимизации, компьютерная алгебра

Basics of solving optimization problems on graphs in the Wolfram Mathematica computer algebra package

Osipov Gennadij Sergeevich
Sakhalin State University
Doctor of technical Sciences, Head of the Department of Computer Science

Vashakidze Natella Semenovna
Sakhalin State University
Associate Professor, Department of Computer Science

Filippova Galina Viktorovna
Sakhalin State University
Associate Professor, Department of Computer Science

Abstract
The formal formulation of the most well-known optimization problems on graphs is presented - the problem of the shortest path, the traveling salesman problem and the maximum flow. Practical testing of problem solving methods was carried out in

the Wolfram Mathematica package environment, which is one of the most powerful information processing systems using symbolic mathematics and computer algebra.

Keywords: graphs, optimization algorithms, computer algebra

Введение

Целью настоящего исследования является комплексное исследование наиболее известных задач оптимизации на графах [1] как совокупности их формальной математической постановки и практической реализации в одной из наиболее развитых систем компьютерной математики *Wolfram Mathematica* [2, 3]. Задачи о кратчайшем пути в графе, проблема коммивояжера и отыскание максимального потока в сети являются классическими, которые служат методологическим объектом для отработки (апробации) современных методов оптимизации и одновременно имеют широкий спектр практического применения в различных предметных областях.

1 Задача о кратчайшем пути

Маршрутом μ в графе $G=(V,E)$ называется такая последовательность $v_1e_1v_2e_2...v_ke_kv_{k+1}$, где $v_i \in V$, $e_i \in E$, что для каждого $i=\overline{1,k}$ ребро e_i соединяет вершину v_i с вершиной v_{i+1}. При этом принято говорить, что маршрут μ соединяет вершины v_1 и v_{k+1} и называть такой маршрут (v_1,v_{k+1})-маршрутом.

Маршрут называется *путем*, если все его элементы, кроме, возможно, v_1 и v_{k+1}, различны.

Путь, в котором $v_1=v_{k+1}$, называется *циклом*.

Граф называется *связным*, если для каждых двух его различных вершин u,v существует (u,v)-маршрут.

Пусть $G=(V,E)$ – ориентированный граф . Говорят, что вершина $v \in V$ *достижима* из вершины $u \in V$, если существует некоторый (u,v)-маршрут.

Весом маршрута называется сумма весов всех входящих в него ребер.

Предположим, что вершина $v \in V$ достижима из вершины $u \in V$, и рассмотрим множество $M(u,v)$ всех (u,v)-маршрутов в G.

Маршрут $\mu \in M(u,v)$ называется *кратчайшим путем*, если его вес не превосходит веса любого другого маршрута из $M(u,v)$.

Задача поиска кратчайшего пути формулируется следующим образом:

дан граф $G=(V,E)$, каждой дуге которого приписан некоторый «вес» (стоимость).

требуется найти кратчайший путь из вершины $u \in V$ в вершину $v \in V$.

В пакете Wolfram Mathematica (WM) существует встроенная функция *Dijkstra*[<взвешенный граф>, <вершина>], которая ищет кратчайшие пути из заданной вершины во все остальные и выдает дерево, составленное из этих

кратчайших путей, а также список весов этих путей [2]. Если вершина недостижима, вес пути равен ∞. Дерево задано списком, на *i*-м месте которого стоит номер предшественника *i*-й вершины на ее кратчайшем пути. Алгоритм Дейкстры можно использовать только в случае, если все веса неотрицательны.

Функция *BellmanFord*[<взвешенный граф>, <вершина>] определяет кратчайший путь по алгоритму Беллмана-Форда. Алгоритм Беллмана-Форда может использоваться в случае, если в графе присутствуют ребра с отрицательными весами.

Функция *ShortestPath* [<граф>, <начальная вершина>, <конечная вершина>], которая позволяет определить кратчайший путь в графе и выдает результат в виде списка последовательных вершин.

Этапы решения задачи в WM (пример):

1. задание координат вершин планарного графа;

$v = \{\{-1,0\},\{-0.5,0.8\},\{0.5,0.8\},\{1,0\},\{0.5,-0.8\},\{-0.5,-0.8\}\}$;

2. построение графа из упорядоченных пар вершин заданного множества *v*;

$g = FromOrderedPairs\left[\{\{1,2\},\{2,3\},\{3,4\},\{5,4\},\{6,5\},\{1,6\},\{2,5\}\},v\right]$;

3. расстановка весов ребер;

$w = \{2,1,5,1,2,3,4\}$;

4. задание меток ребер, равных их весам;

$g = SetEdgeLabels[g,w]$;

5. изображение графа (см. рис. 1);

$ShowGraph[g, EdgeLabelColor \to Blue, VertexLabel \to True]$;

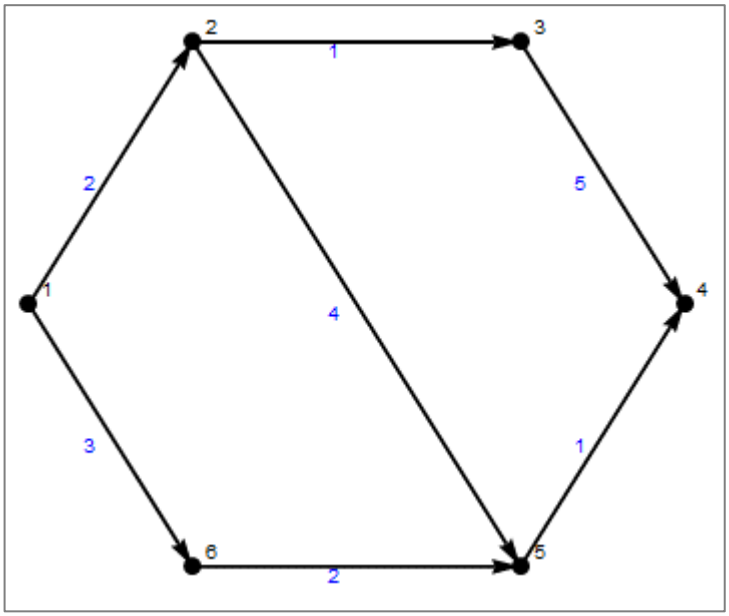

Рис. 1 Схема исследуемого графа

6. получение решения по алгоритму Дейкстры или Беллмана–Форда;

Dijkstra[*g*, 1] (*BellmanFord*[*g*, 1])→{1, 1, 2, 5, 6, 1}, {0, 2, 3, 6, 5, 3}};

7. нахождение кратчайшего пути из 1 в 4 вершину;

SP=ShortestPath[*g*,1,4]→{1, 6, 5, 4};

8. отображение результата (см. рис. 2);

ShowGraph[*Highlight*[g,{*Partition*[*SP*,2,1]},*HighlightedEdgeColors-> Green, VertexLabel -> True, HighlightedEdgeStyle -> Thick*]].

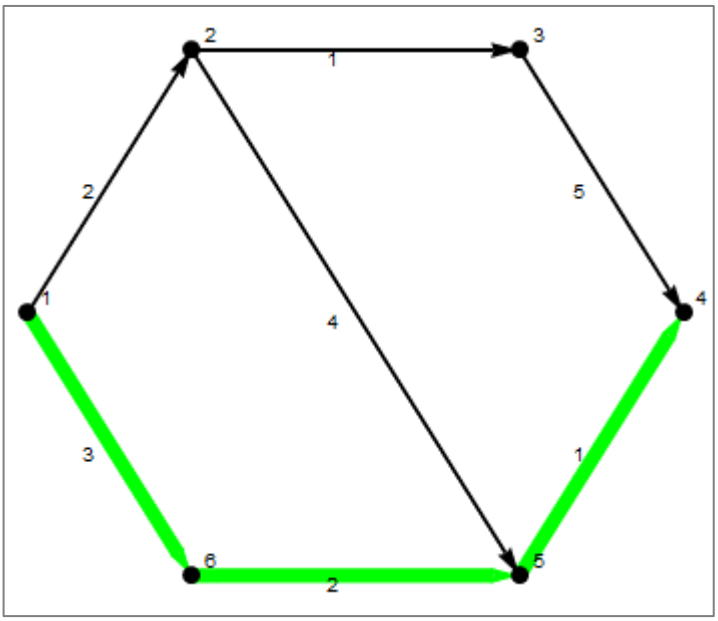

Рис. 2 Кратчайший путь из 1 в 4 вершину

2 Задача коммивояжера

Имеется сеть из n городов, связанных между собой транспортными коммуникациями.

Задана матрица $C=\left(c_{ij}\right)_{i=\overline{1,n}}^{j=\overline{1,n}}$, где элементы c_{ij} определяют затраты (стоимостные, временные) на переезд между i-ым и j-ым городом.

Коммивояжер, выехав из исходного города, должен объехать все города, посетив каждый из них только один раз, и вернуться в исходный.

Требуется определить, в каком порядке следует объезжать города, чтобы суммарные затраты были минимальными.

Если затраты на переезд между парой городов не зависят от направления движения, то задача называется *симметричной*, в противном случае – *несимметричной*.

Задача коммивояжера одна из наиболее известных комбинаторных оптимизационных задач. Она является базовой моделью для апробации новых методов оптимизации и имеет широкий спектр практического применения в различных предметных областях.

В терминах Теории графов задача коммивояжера сводится к поиску кратчайшего гамильтонова цикла в полном графе.

В WM есть функция *TravelingSalesman*, которая позволяет определить оптимальный путь (тур) коммивояжера в неориентированном графе.

Функция *CostOfPath*[*g*, *p*] суммирует веса ребер пути p в графе g, т.е. с ее помощью можно вычислить общую стоимость пути.

Этапы решения задачи в WM:

1. задаем матрицу весов $c = \begin{pmatrix} \infty & 1 & 10 & 7 & 6 & 9 \\ 1 & \infty & 9 & 3 & 5 & 2 \\ 10 & 9 & \infty & 5 & 3 & 10 \\ 7 & 3 & 5 & \infty & 6 & 4 \\ 6 & 5 & 3 & 6 & \infty & 3 \\ 9 & 2 & 10 & 4 & 3 & \infty \end{pmatrix}$;

2. конструируем граф;

$g = FromAdjacencyMatrix[c, EdgeWeight]$;

3. присваиваем меткам ребер значения их весов;

$g = SetEdgeLabels[g, \#[[2]] \& /@ Edges[g, EdgeWeight]]$;

4. отображаем граф (см. рис. 3);

$ShowLabeledGraph[g, EdgeLabelColor \to Blue, VertexColor \to Green]$

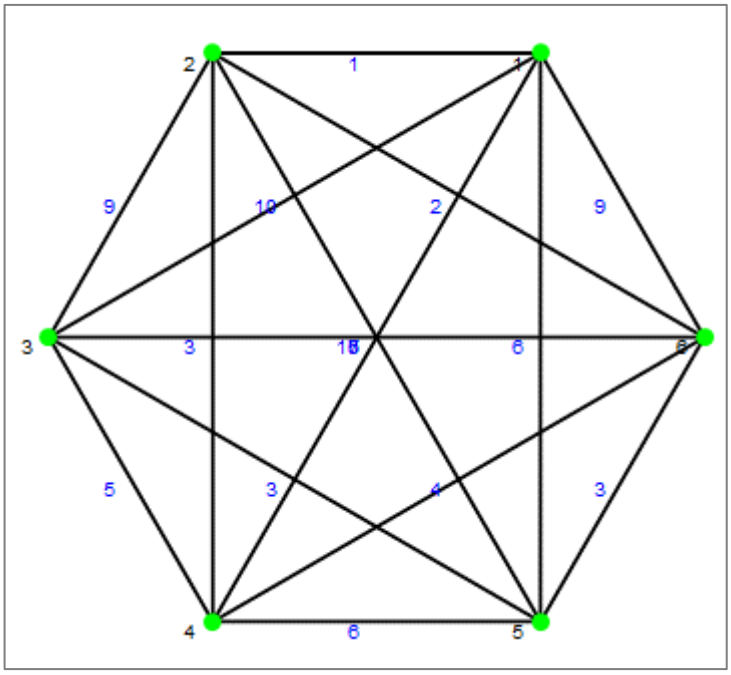

Рис. 3 Схема сети городов

5. решаем задачу коммивояжера;

$TSP = TravelingSalesman[g] \to \{1, 2, 6, 4, 3, 5, 1\}$;

6. вычисляем стоимость оптимального пути;

$CostOfPath[g, TSP] \to 21$;

7. задаем матрицу весов для оптимального пути;

$$cres = \begin{pmatrix} \infty & 1 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & 3 & \infty \\ \infty & \infty & 5 & \infty & \infty & \infty \\ 6 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 4 & \infty & \infty \end{pmatrix};$$

8. конструируем и отображаем граф для визуализации оптимального пути;

$$gres = FromAdjacencyMatrix[cres, EdgeWeight, Type \to Directed];$$

9. устанавливаем метки ребер, равные их весам;

$$gres = SetEdgeLabels[gres, \#[[2]] \& / @ Edges[gres, EdgeWeight]];$$

10. отображаем граф (см. рис. 4)

$$ShowLabeledGraph[gres, EdgeLabel \to True, EdgeLabelColor \to Blue].$$

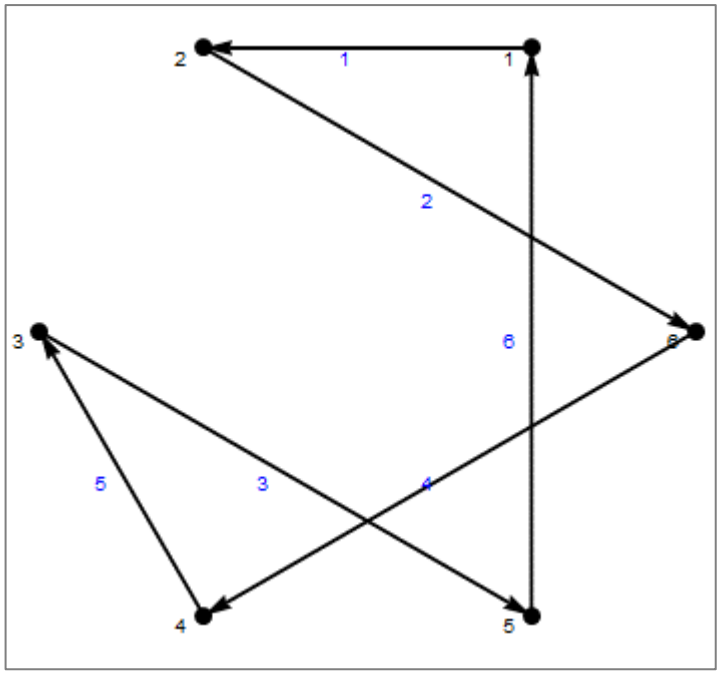

Рис. 4 Графическое представление тура коммивояжера

3 Задача о максимальном потоке

Любой реберно-взвешенный граф $G = (V, E)$ можно рассматривать как сеть труб (или других коммуникаций), причем вес ребра $G = (V, E)$ задает его пропускную способность. $G = (V, E)$, где V

Задача о максимальном потоке заключается в нахождении максимально возможного потока от $u \in V$ к $v \in V$, удовлетворяющего ограничениям на пропускную способность каждого ребра.

Функция *NetworkFlow*[*g*, *source*, *sink*] возвращает значение максимального потока в графе *g* из *source* в *sink* (используется метод Форда-Фалкерсона).

NetworkFlow[*g*, *source*, *sink*, *Edge*] возвращает ребра в *g*, которые имеют положительный поток в максимальном потоке из *source* в *sink*.

Пусть известна топология графа g (рис. 5).

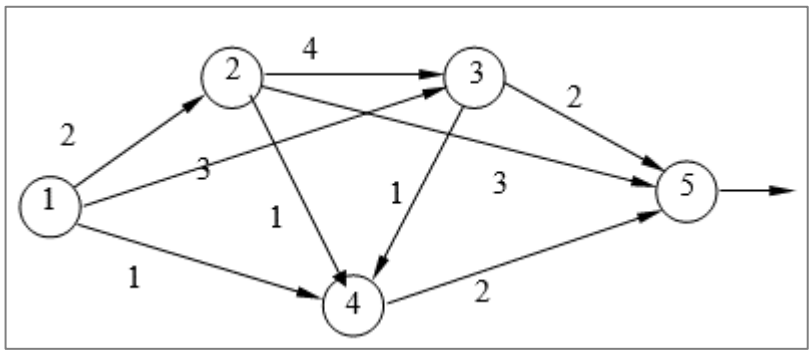

Рис. 5 Пример графа для исследования

Этапы решения задачи в WM:

1. задание матрицы весов;

$$c = \begin{pmatrix} \infty & 2 & 3 & 1 & \infty \\ \infty & \infty & 4 & 1 & 3 \\ \infty & \infty & \infty & 1 & 2 \\ \infty & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \end{pmatrix};$$

2. построение графа на основе матрицы смежности;

$$g = FromAdjacencyMatrix[c, EdgeWeight, Type \to Directed];$$

3. присвоение меткам ребер значения весов;

$$g = SetEdgeLabels[g, GetEdgeWeights[g]];$$

4. изображение графа (см. рис. 6);

$$ShowLabeledGraph[g, EdgeLabel \to True, EdgeLabelColor \to Blue]$$

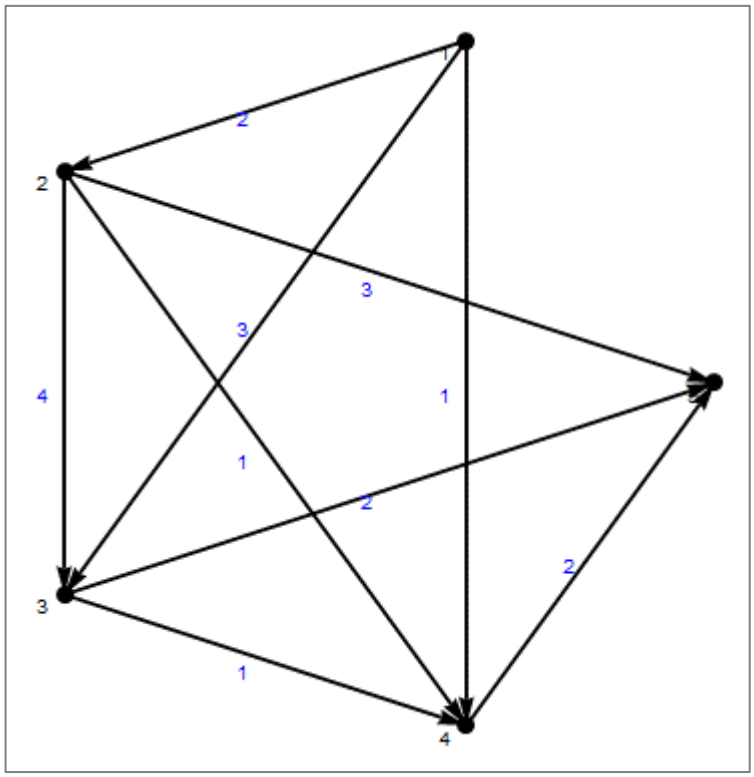

Рис. 6 Изображение исходного ориентированного графа

5. Решение задачи о максимальном потоке;

$NetworkFlow[g,1,5] \to 6$;

6. Потоки по коммуникациям;

$NetworkFlow[g,1,5,Edge]$;

$\{\{\{1,2\},2\},\{\{1,3\},3\},\{\{1,4\},1\},\{\{2,5\},2\},\{\{3,4\},1\},\{\{3,5\},2\},\{\{4,5\},2\}\}$.

Заключение

Проведенное исследование позволило свести в единую систему формальную строгость математической постановки классических задач оптимизации на графах и современный аппарат символьных вычислений, реализованный в пакете компьютерной алгебры *Wolfram Mathematica*.

Предложенная методология синтеза совокупности формальных процедуральных знаний и возможностей исчисления символов является унифицированной и может быть применена для решения большого класса задач с различной семантической интерпретацией.

Библиографический список

1. Иванов А., Ильютко Д., Носовский Г. и др. Графы в компьютерной геометрии // Практикум по компьютерной геометрии. Лекционный материал НОУ "Интуит". URL: https://www.intuit.ru/studies/professional_skill_improvements/1845/info. Дата обращения 15.03.2019.
2. Бурзалова Т.В. Учебно-методический комплекс по решению задач дискретной математики с использованием компьютерной системы «Mathematica». Улан-Удэ: Издательство Бурятского госуниверситета, 2002. 300 с.
3. Половко А.М. Mathematica для студента. СПб: БХВ-Петербург, 2007. 368 с.