Анализ применения классических алгоритмов для поиска кратчайшего пути на графе применительно к геоинформационным системам

Смирнов Сергей Владимирович, кандидат технических наук, старший научный сотрудник
ИПУ РАН (г. Москва)

Аннотация. *В докладе проведён анализ классических алгоритмов поиска кратчайшего пути в геоинформационных системах (ГИС): Форда-Фалкерсона, маршрутного и волнового. Представлено обоснование преимущества использования сетевого анализа при решении некоторых задач. Показаны положительные и отрицательные моменты применения, рассматриваемых алгоритмов для построения маршрутов на электронной карте ГИС.*

Ключевые слова: *поиск кратчайшего пути, электронная карта, геоинформационные системы (ГИС), компьютерные игры, сетевой анализ, алгоритм Форда-Фалкерсона, маршрутный алгоритм, волновой алгоритм, построение трассы, фронт волны, элемент.*

DOI: 10.5281/zenodo.5806408

Введение

Для нахождения кратчайшего пути между объектами в различных сферах жизнедеятельности человека необходимо решить один из классов задач, относящийся к сетевым. Сетевые задачи могут быть сформулированы в терминах линейного программирования. Сетевые задачи, в которых описываются решения по поиску кратчайших путей и максимального пропускного потока, являются транспортными [1].

Сетевые задачи применяют при проектировании и совершенствовании больших и сложных систем, а также при поиске путей их наиболее рационального использования. В первую очередь это связано с тем, что с помощью сетей можно довольно просто построить модель системы. Преимущества применения сетевых моделей также включают в себя следующее: описание многих реально существующих систем, более простое понимание, поиск оптимизационных решений при анализе больших систем.

Сетевые задачи применяют при проектировании и совершенствовании больших и сложных систем, а также при поиске путей их рационального использования. Это связано с тем, что с помощью сетей можно просто построить модель системы. Кроме того, они представляют возможность к более простому пониманию и позволяют находить оптимизационные решения при анализе больших систем [2].

Сетевой анализ берёт своё начало с известной задачи выдающегося немецкого математика Эйлера о кёнигсбергских мостах. Через столетие известные учёные в области физики Джеймс Клерк Максвелл и Густав Роберт Кирхгофф, исследуя электрические цепи, сформулировали некоторые основные принципы сетевого анализа. В начале XX века европейскими и американскими инженерами были разработаны методы расчёта наибольшей пропускной способности телефонных линий и коммутаторов, позволяющих обеспечивать гарантированное обслуживание определённого числа абонентов. В связи с развитием вычислительной техники стали проводиться более глубокие исследования построенных моделей и поиска путей.

Благодаря специальной структуре сетевых задач для них получено большое число эффективных алгоритмов, обеспечивающих решение практических задач. Для решения задачи поиска кратчайших путей между объектами возможно применение нескольких алгоритмов. Существует определённая проблема при автоматизации извлечения геоинформации при применении различных алгоритмов поиска пути. Одни алгоритмы обеспечивают скорый поиск, не загружая оперативную память компьютера громоздкими вычислениями, а другие, наоборот, требуют больше математических операций. Рассмотрим те алгоритмы, которые позволят решить задачи и не создадут проблем при автоматизации извлечения информации.

Существуют различные постановки задачи о кратчайшем пути: между заданной парой, в заданный пункт назначения, о кратчайшем пути вершин, о кратчайшем пути между всеми парами вершин.

В различных постановках задачи, роль длины ребра могут играть не только сами длины, но и время, стоимость, расходы, объём затрачиваемых ресурсов или другие характеристики, связанные с прохождением каждого ребра. Таким образом, задача находит практическое применение в большом количестве областей (информатика, экономика, география и др.) [3].

В связи с тем, что существует множество различных постановок этой задачи, есть наиболее популярные алгоритмы для поиска кратчайшего пути на графе: *Дейкстры*, *Форда-Фалкерсона* , *Беллмана — Форда*, *Джонсона*, *Ли (волновой)*, *Килдала*, *маршрутный* и т.д.. Алгоритмы нахождения кратчайшего пути на графе применяются для нахождения путей между физическими объектами на таких картографических сервисах, как карты Google или OpenStreetMap [4].

Остановимся на алгоритмах, которые можно применить к геоинформационным системам и проанализируем их: Форда-Фалкерсона, волновой и маршрутный.

Алгоритм Форда - Фалкерсона

Алгоритм Форда - Фалкерсона известен как метод расстановки пометок и основан на следующем простом факте, который гласит, что если имеется кратчайший путь между какими-либо двумя вершинами, то его часть между любыми двумя вершинами на нём же самом также является кратчайшим путём. Алгоритм позволяет найти кратчайшие пути из какой-либо одной вершины графа во все остальные.

Исходную вершину можно выбрать произвольно, но, сделав выбор, далее исходить из того, что будут описаны кратчайшие пути именно из этой вершины во все остальные. Идея, лежащая в основе этого алгоритма, состоит в следующем. Выбирается начальный поток из s в t и с помощью алгоритма поиска увеличивающей цепи выполняется поиск следующей цепи и т.д. [5].

Рассмотрим данный алгоритм более детально:

1). Пусть имеется взвешенный граф G=[A, B], в котором пронумеруем все вершины, так что $A=\{1,2,...,p\}$, но при этом номер «1» имеет именно та вершина, из которой будут найдены кратчайшие пути во все остальные вершины. Построим далее матрицу $M=(m_{ij}),где \quad i,j=1,2,...,p$, приняв во внимание следующую условность:

$$m_{ij}=\begin{cases} f((i,j)), & если \quad (i,j)\in B, \\ клетка \quad остается \quad незаполненой, & если \quad (i,j)\notin B. \end{cases}$$

№ №	1	2	3	4	5	6	7	8
1		11	14					
2	11						12	
3	14			18	11			
4			18			17	13	
5			11			12		16
6				17	12		16	11
7		12		13		16		
8					16	11		

2). Около первой строки матрицы M, слева от матрицы, поставим числовую пометку «0» и такую же пометку проставим над первым столбцом матрицы. Затем просмотрим помеченную строку слева направо и всякий раз, встречая клетку с числом, прибавим это число к пометке строки и сумму проставим над столбцом, в котором эта клетка находится.

№	1	2	3	4	5	6	7	8	9
1		0	11	14	31	24	47	22	39
2	0		11	14					
3	11	11						12	
4	14	14			18	11			
5	31			18			17	13	
6	24			11			12		16
7	47				17	12		16	11
8	22		12		13		16		
9	39					16	11		

3). После этого отразим пометки столбцов относительно главной диагонали. Возникнут помеченные строки. С каждой из помеченных строк проделаем то же самое: просмотрим помеченную строку слева направо и всякий раз, встречая клетку с числом, прибавим это число к пометке строки и сумму проставим в качестве пометки над столбцом, в котором она находится.

4). Затем пометки столбцов отразим относительно главной диагонали и с помеченными строками проделаем то же самое. И так далее, пока не окажутся помеченными все строки и все столбцы. Просмотрим строки таблицы в порядке возрастания их номеров. В каждой строке просматриваются клетки слева направо и всякий раз, когда встречается число, оно складывается с пометкой строки и сумма сравнивается с пометкой столбца, в котором найденное число расположено.

5). Если сумма оказалась меньше, чем пометка столбца, то эта пометка столбца заменяется упомянутой суммой. Если же сумма оказалась больше или равной пометке, то ничего не меняется. После такого просмотра всех строк новые пометки столбцов отражаются относительно главной диагонали, и вся процедура повторяется. И так до тех пор, пока не прекратятся изменения в пометках. После этого по пометкам можно построить кратчайшие пути из первой вершины во все остальные.

путь	длина пути	путь (от конца к началу)
1⇒2	11	2⇐1
1⇒3	14	3⇐1
1⇒4	31	4⇐ 3⇐1
1⇒5	24	5⇐3⇐1
1⇒6	35	6⇐7⇐2⇐1
1⇒7	22	7⇐2⇐1
1⇒8	39	8⇐5⇐3⇐1

В завершении отметим важное свойство алгоритма: если пропускные способности всех дуг являются целыми числами, то, как максимальный поток, так и все промежуточные потоки в алгоритме Форда-Фалкерсона, являются целочисленными. Заметим, что этот алгоритм имеет значительную вычислительную сложность по сравнению маршрутным и волновым.

Маршрутный алгоритм

Маршрутный алгоритм получил свое название, вследствие того, что осуществляет одновременно формирование фронта волны и прокладывание трассы. Основное употребление имеет в радиоэлектронике для нахождения кратчайшего пути без пересечения множества занятых и запрещённых элементов (участков печатной платы). Однако имеет применение и на электронных картах, как один из трассировщиков пути. К слову сказать, наибольшее применение этот алгоритм имеет в компьютерных играх при разработке стратегии по электронным картам. Прокладка пути с помощью маршрутного алгоритма похожа на прокладывание пути с помощью волнового алгоритма, где источником волны на каждом шаге является конечный элемент участка трассы проложенной на предыдущих шагах. Алгоритм реализует следующие последовательно выполняемые этапы: построение пути до встречи с препятствием, обход препятствий и минимизация построенного пути.

Маршрутный алгоритм имеет две разновидности, которые существенно отличаются друг от друга при построении трассы: *алгоритм, основанный на вычислении расстояния между точками* и *на рекуррентном соотношении.*

Алгоритм, основанный на вычислении расстояния между точками. В этом случае работа алгоритма начинается от начального элемента. При этом вокруг

начального элемента рассматривается 8-ми элементная окрестность. От каждого элемента окрестности до конечного элемента оценивается длина пути. При этом расстояние между точками вычисляется по формуле [6]:

$$D=|X_i - XB|+ |Y_i - YB|,$$

где (X_i,Y_i) - координаты точки окрестности; (XB,YB) - координаты конечного элемента.

В результате вычисляем восемь значений, из которых и выбираем минимальное. Элемент, расстояние от которого оказалось минимальным, выбирается в качестве элемента трассы. Вокруг него снова рассматривается 8-ми элементная окрестность. Процесс продолжается до тех пор, пока не будет достигнут конечный элемент. Если на пути встречается препятствие в виде запрещенного элемента, то обход препятствия осуществляется, исходя из интуиции разработчика.

Алгоритм, основанный на рекуррентном соотношении. В данном случае построение осуществляется на основе рекуррентного соотношения, представленного с помощью формулы: $y(x) = 2y(x + h) + y(x + 2h) + d$,

где x, $y(x)$ - абсцисса и ордината элемента, занимаемого трассой на

шаге;

$(x + h)$ - ордината элемента, занимаемого трассой на предыдущем шаге;

$(x + 2h)$ - ордината элемента, отстоящего от вычисляемого на 2 шага;

h - величина изменения абсциссы на каждом шаге;

d (delta) - функция, определяющая вид трассы.

Ордината очередного элемента трассы вычисляется по рекуррентной формуле (формуле приведения). Формула приведения помогает связать значения p +1 соседних элементов $(x + h)_k$, $(x + h)_{k-1}$, ..., $(x + h)_{k-p}$ (где $k \geq p+1$), некоторой последовательности $\{(x + h)_n\}$ (где $n=1,2,\ldots$):

$(x + h)_k=f(k, (x + h)_{k-1},\ldots, (x + h)_{k-p})$

Рекуррентная формула позволяет определить любой элемент последовательности, если известны p первых её элементов $(x + h)_1$,

$(x + h)_2,\ldots, (x + h)_p$.

Абсцисса трассы вычисляется по формуле: $D=X_n=X_{n-1} + h$

Знак « + »или « - » в рекуррентной формуле выбирается исходя из того, откуда начинается построение трассы, из начального элемента « + », и соответственно из конечного « - ».

По этой формуле, чтобы вычислить 3-й элемент трассы, необходимо знать два предыдущих. Первым элементом является исходный элемент A(XA,YA), тогда ордината второго элемента вычисляется по формуле:

$Y(X)=Y(XA)+ ((Y(XA)-Y(XB))/(XA-XB))*h$

Если на пути встретится запрещенный элемент, то его обход осуществляется исходя из интуиции разработчика.

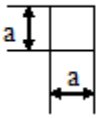
Главным достоинством маршрутного алгоритма является простота, а также возможность движения по диагонали. Несомненно, данный алгоритм подходит для применения при трассировке и поиске кратчайшего пути на карте в силу его простой реализации и небольшого количества вычислений. Данный алгоритм отличает манёвренность, но зачастую он не функционирует в некоторых ситуациях (сложная дорожная сеть) и даёт сбои в задаче трассировки.

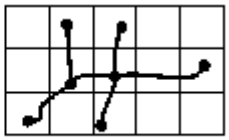
Волновой алгоритм (алгоритм Ли)

В завершении обзора рассмотрим *волновой алгоритм (алгоритм Ли), который* является одним из самых универсальных алгоритмов трассировки. Представляет собой оптимальный вариант для трассировки печатных соединений в радиоэлектронике. Однако и он с недавних пор применяется для построения трасс на географических электронных картах, позволяя построить трассу (путь) между двумя элементами в любой точке. Применяя волновой алгоритм, можно добиться большего результата чем от рассмотренных выше, но для применения на электронных картах требует небольшой доработки, в частности, построение трассы по диагонали.

Задача нахождения самого короткого пути между точками на карте с произвольно расположенными препятствиями характерна, в первую очередь, для индустрии компьютерных игр. Однако, волновой алгоритм можно также применить для поиска пути на электронной карте геоинформационной системы. С помощью него можно решить довольно сложные задачи, например, построение оптимальных маршрутов для инспектирования, нахождение кратчайших путей между объектами и т.д.

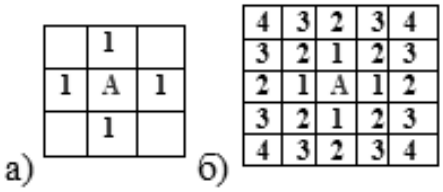
Объясним действие алгоритма более подробно. Пусть имеется рабочее поле (матрица смежности), которое разбивается на элементарные ячейки. Размеры ячеек и их количество определяются площадью поля, допустимой плотностью расположения объектов и дорожной сетью. Выбранная система ячеек определяет среду, в которой осуществляется построение соединений. В данном случае ячейка представляет собой квадрат со стороной a=32 ед. [7-8]





Если размеры поля по горизонтали и вертикали соответственно A_x и B_y, то получим дискретное рабочее поле (ДРП) с N_x*N_y ячейками, где $N_x=\{A_x/a\}$, $N_y=\{A_y/a\}$, причём {} – символы ближайшего большего целого.

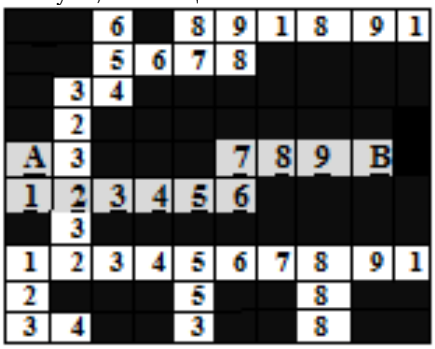
Основу всех модификаций алгоритма Ли составляет процедура построения оптимального в заданном варианте пути между двумя известными ячейками дискретного рабочего поля. Процедура состоит из двух этапов: поиска пути и проведения пути. Из начального элемента распространяется в 4-х направлениях волна (рис. а). Элемент, в который пришла волна, образует фронт волны. На рисунках цифрами обозначены номера фронтов волны. Каждый элемент первого фронта волны является источником вторичной волны (рис б).



Элементы второго фронта волны генерируют волну третьего фронта и т.д. Процесс продолжается до тех пор пока не будет достигнут конечный элемент.

На втором этапе строится сама трасса. Её построение осуществляется в соответствии со следующими правилами: движение при построении трассы осуществляется в соответствии с выбранными приоритетами, при движении от конечного элемента к начальному номер фронта волны должен уменьшаться, приоритеты направления движения выбираются на стадии разработки, в зависимости от приоритета направления получаются разные трассы, но длина трассы в любом случае остаётся одной и той же.

Рассмотрим небольшой пример применения волнового алгоритма. Условимся, что чёрным цветом будут отмечены запрещенные элементы (это могут быть жилые кварталы, тротуары и т.д.), а серым цветом обозначим трассу после действия алгоритма. Заметим также, что А-начальная и В-конечная точки пути. Приоритеты маршрутного движения будут влево, вправо, вверх, вниз, однако при доработке алгоритма возможно движение и по диагонали. Движение по диагонали аналогично движению влево, вправо, вверх, вниз, но с углом наклона α=45°. Построение трассы ведется от начальной точки к конечной. Приоритетные направления показаны серым цветом и подчёркнутыми буквами и цифрами, где А начало пути, В конец.



Преимущества волнового алгоритма в том, что с его помощью можно найти трассу в любом дорожном лабиринте и с любым количеством запретных элементов (квартал, тротуар и т.д). На электронных картах довольно часто применяют маршрутный алгоритм (или его модификации) в силу его простой реализации и небольшого количества вычислений. Но он часто не пригоден в сложной дорожной ситуации (сложная дорожная развязка). [9]

Вычислений при применении волнового алгоритма меньше чем в других, рассмотренных нами алгоритмах, ведь номер фронта волны уже и есть расстояние до начального элемента, значит можно просто выбрать наименьший элемент из окрестности и продвинуть трассу в его сторону. Недостатком алгоритма является большие потребности памяти для хранения вспомогательной карты фронтов волн. Однако эта проблема практически незаметна на современных компьютерах с большим объемом памяти.

Заключение

Исходя из рассмотренных выше традиционных алгоритмов, приходим к выводу, что оптимальным алгоритмом для применения при решении задачи поиска кратчайшего пути в ГИС является волновой. При применении волнового алгоритма не составляет сложности автоматизация извлечения геоинформации при решении массовых задач, какими являются задачи поиска кратчайшего пути для инспектирования объектов в различных сферах, где необходима информационная поддержка.

Разумеется, что кроме волнового, существует сравнительно большее количество методов для поиска маршрутов. Где-то требуется наибольшая скорость расчётов в ущерб качеству, иногда наименьшее число поворотов, а местами необходимо, чтобы маршрут обязательно прошёл через некоторые ключевые точки.

Литература:

1. Книжников Ю.Ф., Кравцова В.И. Новые аспекты традиционной картографической проблемы генерализации// Взаимодействие картографии и геоинформатики (к 60-летию профессора С.Н. Сербенюка) /Под. ред. А.М.Берлянта и О.Р.Мусина. -М.: Научный мир, 2000.-192 с.
2. Смирнов С.В., Тюкавкин Д.В. Геоинформационная система для поддержки принятия решений в органах управления социально-образовательной сферой // Проблемы управления. – М.: Изд-во «СенСиДат», 2003. №3. – с.54-60.
3. Задача о кратчайшем пути / / URL: https://ru.wikipedia.org/wiki/Задача_о_кратчайшем_пути; (дата обращения: 17.12.2021).
4. Sergei Smirnov, Liudmila Sizova, «Application of EDA-system to find the shortest path on an electronic GIS map», CPT2021 Computing for Physics and Technology. The 9th International Conference on Computing for Physics and Technology (CPT2021). Conference Proceedings (2021), Nizhny Novgorod – Moscow – Pushchino, Russia, November 08-12, 2021, p. 82-90.

5. Смирнов С.В. Исследование возможностей применения алгоритмов определения кратчайшего пути в ГИС / Труды 4-й Международной научно-практической конференции "АКТУАЛЬНЫЕ ВОПРОСЫ СОВРЕМЕННОЙ НАУКИ: ТЕОРИЯ, МЕТОДОЛОГИЯ, ПРАКТИКА, ИННОВАТИКА" (Уфа, 2020). Уфа: НИЦ Вестник науки, 2020. С. 56-61.
6. Нахождение пути на карте / / URL: https://firststeps.ru/theory/karta.html; (дата обращения: 17.12.2021).
7. Евстигнеев В.А. Теория графов и программирование. – Новосибирск: Изд-во НГУ, 1978.
8. Селютин В.А. Машинное конструирование электронных устройств. – М.: «Сов. радио», 1977. – 384 с.
9. Смирнов С.В. Проектирование графических систем со сложной структурой данных. - Саарбрюкен: LAP Lambert Academic Publishing, 2011. – 176 с.