Система построения маршрутов на пересеченной местности на основе графа видимости

Д.В. Козуб[1], Ю.С. Корухова[1]

[1] *Московский государственный университет им. М.В. Ломоносова*

Аннотация. В работе представлен подход к решению задачи построения маршрутов на пересеченной местности и дискретизации географического ландшафта с помощью графа видимости. Традиционно системы построения маршрутов находят путь по графу дорог, однако задача поиска маршрутов, проходящих вне дорог, также является актуальной. Предлагается применить метод аппроксимации многоугольников к построению графа видимости на плоскости, а также метод поиска кратчайшего пути без построения полного графа для решения задачи навигации на пересеченной местности. Описывается разработанный алгоритм построения опорных прямых к выпуклому многоугольнику на плоскости и иерархический подход к построению графа видимости. Предложенные подходы реализованы в программной системе и применены к реальным географическим данным.

Ключевые слова: навигационная система, вычислительная геометрия, граф видимости

Off-road routing system based on visibility graph

D.V. Kozub[1], Y.S. Korukhova[1]

[1] *Lomonosov Moscow State University*

Abstract. The problem of off-road routing and terrain discretization with using visibility graphs is considered. Traditionally, routing engines find a path over a road graph, but the problem of off-road routing is also relevant. A polygon approximation method is proposed as applied to constructing a visibility graph on a plane, as well as pathfinding without constructing a complete graph for solving the problem of off-road navigation. The developed algorithm for finding supporting lines to a convex polygon on a plane and a hierarchical approach to building a visibility graph are described. The considered approaches are implemented in a software system and applied to real geographic data.

Keywords: navigation system, computational geometry, visibility graph

1. Введение

В настоящее время значительное количество информации хранится в электронном виде. В том числе существуют различные электронные карты (Google Карты, Яндекс Карты, OpenStreetMap) и средства работы с ними. Одной из наиболее актуальных задач является построение маршрутов, однако существующие на сегодняшний день картографические и навигационные сервисы предоставляют возможность прокладывать маршрут между заданными точками либо по автодорогам и тротуарам, либо по проселочным и грунтовым проездам. В то же время часто возникает необходимость навигации по бездорожью. Например, при планировании туристических маршрутов и спасательных операций, строительстве автомобильных и железных дорог, прокладке коммуникаций, а также для эффективного использования территориальных и природных ресурсов. В данной работе рассматривается решение задачи построения маршрутов на пересеченной местности с использованием данных электронных карт.

2. Дискретизация местности

Из-за того, что своей сути географические данные являются непрерывными, для удобства их хранения и обработки необходимо дискретизировать. Существуют два основных способа дискретизации географических данных: векторная и растровая графика. В первом случае данные о местности хранятся на плоскости в виде набора многоугольников и ломаных, и каждому из объектов приписаны некоторые атрибуты, описывающие географические свойства. Во втором случае минимальной единицей информации является пиксель, также хранящий атрибутивные данные. Растровые изображения получают с помощью спутниковой съемки, а векторные – путем обработки спутниковых снимков человеком или программой, использующей алгоритмы компьютерного зрения. В данной статье рассматривается векторный формат представления данных.

Отдельно опишем специфику постановки задачи построения маршрутов по пересеченной местности. Будем рассматривать географические данные в двумерном пространстве (положение каждой точки задается парой значений в некоторой системе координат). Объекты (природные или инфраструктуры) на плоскости задаются простыми многоугольниками и ломаными и содержат информацию об их типе. В дальнейших рассуждениях, не ограничивая общности, будем рассматривать лишь множество полигональных объектов $P = \{P_i\}$, $|P| = h$, вершины которых заданы против часовой стрелки.

При решении задачи маршрутизации естественными и наиболее часто используемыми структурами данных являются граф и сетка. Сетка, в случае географических данных, представлена растровым изображением (grid map), и по умолчанию определяет 4 возможных направления движения из каждой ячейки - влево, вниз, вправо, вверх (или 8, если разрешена диагональная

маршрутизация). Граф, в свою очередь, может быть построен несколькими способами по векторным данным.

Первый способ – дорожная карта, основанная на понятии трапецоидной карты [1, стр. 146]. Данный подход часто применяется в робототехнике и компьютерных играх, однако не может быть использован для решения поставленной нами задачи. Он рассматривает многоугольники как непреодолимые препятствия, что в нашем случае не всегда верно и будет зависеть от типа природного объекта.

Второй способ – граф видимости [1, стр. 372]. По определению, для множества многоугольников на плоскости вершинами графа видимости являются вершины многоугольников, а ребро между двумя вершинами существует в случае, если отрезок на плоскости, соединяющий их, не пересекает ни одно из ребер многоугольников. Данное определение может быть тривиальным образом расширено для множества ломаных. Нами будут рассматриваться взвешенные неориентированные графы, вес ребра зависит от типа соответствующего ему природного объекта. Стоит иметь в виду, что для решения задачи, в которой многоугольники представляют собой природные объекты, граф видимости должен быть дополнен всеми диагоналями многоугольников.

3. Построение графа видимости

В 1997 году было доказано, что кратчайший путь между двумя точками на плоскости для множества полигональных препятствий проходит по ребрам графа видимости [1, стр. 375]. Более того, уточняется, что при этом граф видимости может быть редуцированным - его ребра являются общими касательными для пары многоугольников.

Пусть $n = \sum_{i=1}^{h} n_i$ где n_i - кол-во вершин в многоугольнике P_i. Тогда n - общее количество вершин графа видимости. Переборный алгоритм его построения будет иметь оценку временной сложности $O(n^3)$. Алгоритм, использующий принцип заметающей прямой и имеющий временную сложность $O(n^2 \log n)$, приводится в [1, стр. 376]. Для построения графа видимости для набора отрезков на плоскости был предложен алгоритм [2], работающий за время $O(n^2)$, и использующий принципы двойственности отрезков и точек в пространстве. Алгоритм построения редуцированного графа видимости, использующий методы триангуляции, и имеющий временную сложность $O(n + h^2 \log n)$, был предложен в [3].

По сравнению с остальными алгоритмами, метод заметающей прямой обладает несколькими преимуществами: он работает быстро на небольших входных данных, прост в реализации, а также строит ребра видимости для каждой точки запроса q отдельно за время $O(n \log n)$. Последнее свойство представляет особый интерес, так как в случае решения задачи навигации по пересеченной местности построение графа видимости целиком становится вычислительной проблемой. Эвристические алгоритмы поиска

кратчайшего пути на графе в большинстве случаев не проходят по всем ребрам графа, что дает возможность осуществлять поиск кратчайшего пути в графе без его предварительного построения. Для этого необходимо иметь возможность поиска ребер, инцидентных данной вершине - точке запроса q (ребер видимости). Данную возможность и предоставляет алгоритм заметающей прямой. Стоит отметить, что описанный подход также дает возможность реализовать поиск кратчайшего пути в динамически изменяющемся графе (например, при изменении погодных условий или времени суток меняются веса ребер графа) без его полного пересчета.

4. Аппроксимация многоугольников

Нами был предложен алгоритм аппроксимации многоугольников при решении задачи навигации на пересеченной местности с помощью построения графа видимости для множества многоугольников, задающих природные объекты. Введем понятие опорной прямой, тесно связанное с понятием видимости. Прямая называется опорной к фигуре на плоскости, если она имеет с ней общие точки, но не содержит внутренних. На рис. 1 представлены примеры отрезков опорных прямых (q_1l_1, q_1r_1, q_2l_2, q_2r_2).

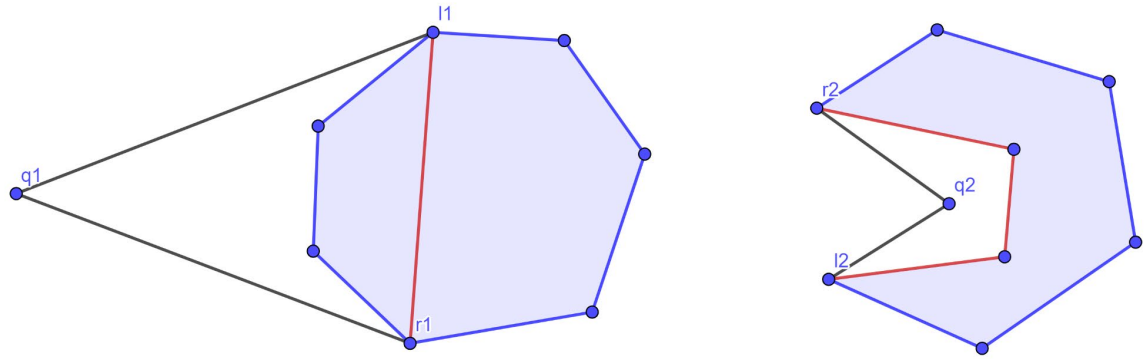

Рис. 1. Отрезки опорных прямых к многоугольникам, проходящие через заданные точки запроса q_1 и q_2.

Пусть в некоторый момент времени необходимо построить ребра графа видимости, инцидентные заданной вершине q. Будем строить опорные прямые, проходящие через q, к каждому из многоугольников P_i. Рассмотрим случай, когда $q \notin \underline{CH}(P_i)$, где $CH(P_i)$ - выпуклая оболочка P_i. В этом случае отрезки опорных прямых ql и qr к P_i будут совпадать с отрезками опорных прямых к $CH(P_i)$. Предобработка многоугольника P_i заключается в построении его выпуклой оболочки, которая может быть произведена за $O(n_i \log k_i)$, где k_i - размер оболочки, построенной с помощью алгоритма Чена [4]. Не ограничивая общности, рассматриваем случай выпуклости P_i, и поставим цель: достичь временной сложности $O(\log n_i)$. Для этого воспользуемся свойством выпуклого многоугольника: полуплоскости, пересечение которых задает многоугольник, отсортированы по индикатору включения в себя любой точки плоскости. В

рассматриваемом случае полуплоскости, однозначно задаваемые парой вершин, разбиваются на два непустых подмножества C и NC (содержащие q и не содержащие q).

Проверка того, является ли прямая опорной к выпуклому многоугольнику, с использованием векторных произведений занимает константное время. Чтобы организовать двоичный поиск по вершинам многоугольника для нахождения опорных точек l и r, необходимо найти две вершины m и M, содержащиеся в C и NC. В 1979 году был предложен алгоритм их поиска за $O(\log n_i)$ [5]. В его основе лежит рекурсивный спуск по модифицированному АВЛ-дереву, в котором хранятся вершины многоугольника, упорядоченные по их полярному углу. На каждой итерации проверяется, являются ли самый левый лист дерева и его корень искомыми точками. Недостатком данного алгоритма является фиксированная структура данных для хранения многоугольника.

Нами был предложен альтернативный алгоритм нахождения точек m и M. Проведем прямую через q и вершину многоугольника m, являющуюся первой в записи вершин многоугольника. Найдем вершину или ребро, в которой данная прямая пересекает многоугольник второй раз (рис. 2). Для этого воспользуемся еще одним свойством выпуклого многоугольника: его вершины упорядочены относительно полярных углов, рассчитанных от некоторой внутренней или граничной точки многоугольника. Предобработка P_i состоит в вычислении данных углов от точки m за линейное от n_i время. Используя двоичный поиск по этим углам, локализуем пересекаемую вершину или ребро. В последнем случае возьмем любую из вершин, задающих ребро. Если найденная вершина совпадает с m, то m - одна из опорных точек, и бинарный поиск организуется по оставшимся вершинам. Иначе мы нашли точки m и M, и бинарный поиск организуется независимо по подмножествам, на которые m и M разбивают вершины многоугольника. Предлагаемый алгоритм имеет логарифмическую сложность.

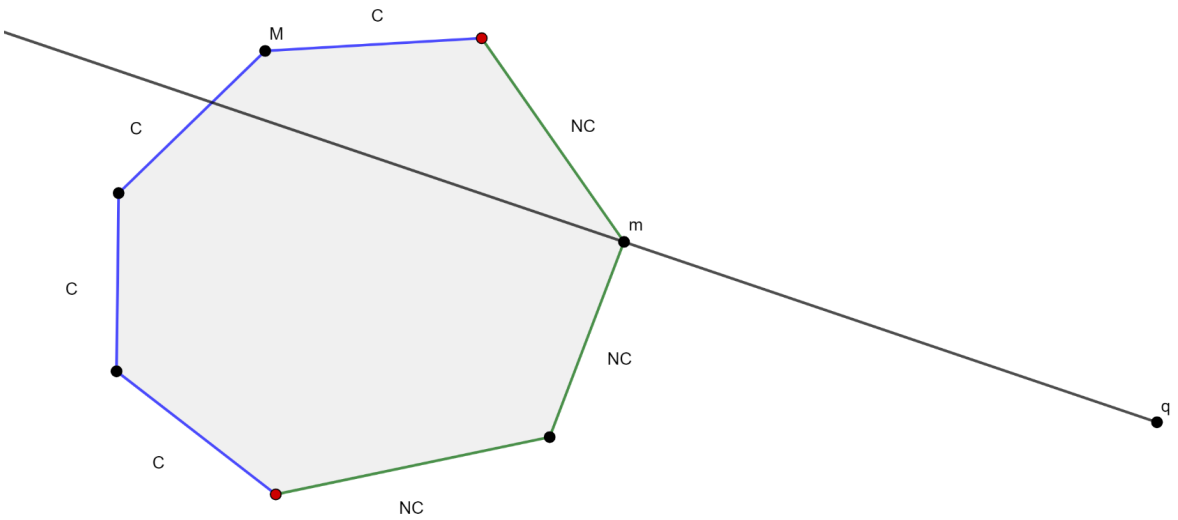

Рис. 2. Метод поиска вершин m и M, содержащихся в C и NC.

В рассмотренном случае аппроксимация многоугольника производится найденным отрезком lr. Для нахождения ребер графа видимости, инцидентных точке запроса q, с помощью алгоритма заметающей прямой за время O(h log h) строятся ребра видимости для q и множества из h отрезков $l_i r_i$, полученных из опорных точек каждого из P_i. Задача локализации q в P_i может быть решена за O(log n_i) операций [6]. Используя неравенство$\sum_{i=1}^{h} \log n_i < h * log(\sum_{i=1}^{h} n_i) = h * \log n$, приходим к оценке временной сложности Ω(h log n) алгоритма построения опорных прямых и Ω(h log nh) алгоритма построения инцидентных ребер.

Если q $\in \underline{CH}(P_i)$, нахождение опорных точек занимает квадратичное время, однако из-за специфики задачи на практике такие случаи крайне редки. При этом многоугольник аппроксимируется не отрезком, а множеством отрезков, соединяющих точки l и r (рис. 1, справа). Следует также отметить, что существуют конфигурации, при которых использование данного подхода не позволяет получить все ребра видимости для множества P корректно, необходимы дополнительные проверки на полное вхождение одного из многоугольников в замыкание выпуклой оболочки другого многоугольника.

Для точки q, которая является вершиной многоугольника P_j, для корректного построения ребер видимости для множества отрезков $l_i r_i$ вводится понятие ограничивающего угла α. Концы отрезков, не попадающие в α, не будут соединены с q ребром. В случае, если q ∈ CH(P_j), α > π задается q и ее соседями по выпуклой оболочке. Иначе, α = ∢ ($l_j q r_j$) ≤ π. Проверка принадлежности точки α с использованием векторных произведений выполняется за константное время.

5. Иерархический подход

Предположим, что нам необходимо построить маршрут длиной в несколько тысяч километров. Граф дорог такой территории может содержать настолько большое количество вершин и ребер, что пользователю придется ждать результата вычислений не один час, в особенности, если вычисления производятся на локальной машине. Для решения данной проблемы на графе дорог применяется иерархический подход. Суть его в следующем: граф дорог делится на локальные кластеры, оптимальное расстояние на преодоление которых вычислено заранее и кэшировано. Поиск кратчайшего пути между кластерами производится с помощью одного из алгоритмов поиска пути на графе. При этом локальные кластеры могут объединяться в кластеры более высокого уровня, так что количество уровней иерархии не ограничено. К примеру, построение маршрута длиной в тысячи километров будет разбито на три части: путь от начальной точки до автомагистрали, путь по автомагистралям, путь от

автомагистрали до конечной точки. Стоит отметить, что построенный маршрут не всегда будет кратчайшим. Иерархический подход к поиску путей также применяется в компьютерных и мобильных играх [7].

В рассматриваемой нами задаче навигации по пересеченной местности количество вершин графа видимости будет в тысячи раз больше, а количество ребер - в миллионы, так как на указанной территории будут рассматриваться не только дороги, но и природные объекты. В этом случае иерархический подход также находит свое применение: отдельными кластерами могут выступать графы видимости на территориях между начальными и конечными точками и графом дорог - именно на этих участках задача навигации по пересеченной местности стоит наиболее остро. При этом в построении всего графа видимости нет необходимости: будет рассматриваться лишь геометрия на указанных участках. Упомянутые два кластера, в свою очередь, также могут быть разбиты на локальные кластеры меньшего размера - в местах, где сконцентрировано наибольшее количество природных объектов. Данное решение может быть использовано для расширения возможностей существующих навигационных сервисов.

В рассмотренном ранее подходе к поиску кратчайшего пути по графу видимости без непосредственного построения всего графа, иерархический подход находит свое место в работе с геометрией природных объектов. В зависимости от размера территории, на которой проводится поиск кратчайших путей, или в зависимости от размера кластеров, на которые разбивается граф видимости, геометрия объектов может быть упрощена. Для упрощения многоугольников и ломаных может быть использован, например, алгоритм Рамера - Дугласа - Пекера [8], для упрощения геометрии в виде графа - алгоритм стягивания ребер, а объекты, слишком маленькие относительно размера территории, могут быть удалены. При этом для поиска кратчайшего пути на более низких уровнях иерархии геометрия объекта восстанавливается.

6. Программная реализация

Описанные подходы к построению маршрутов на пересеченной местности были реализованы в программной системе на языке Python. В качестве географических данных использовались открытые данные OpenStreetMap (www.openstreetmap.org). Программный интерфейс позволяет скачивать, обрабатывать и сохранять географические данные, строить граф видимости, производить построение маршрутов на пересеченной местности, экспортировать и визуализировать результаты. Пример использования системы для навигации по территории Подмосковья представлен на рис. 3. На изображении видно, что маршрут был проложен в обход таких труднопроходимых местностей, как болота и топи (представлены штриховкой). Изначально маршрут идет в противоположном

направлении от цели (Goal), так как таким образом возможно быстрее всего пересечь топь и выйти на дорогу.

Рис. 3. Пример использования программной системы для построения маршрута на пересеченной местности. Слева - спутниковый снимок, справа - данные OpenStreetMap, которые использовала система.

7. Заключение

В работе рассмотрены подходы к решению задачи построения маршрутов на пересеченной местности с помощью графа видимости, предложен способ поиска кратчайшего пути без построения полного графа, а также способ аппроксимации геометрии, позволяющий сократить время решения поставленной задачи. Разобраны способы реализации иерархического подхода к построению маршрутов по пересеченной местности и построению графа видимости, приведен пример использования программной реализации предложенных методов для построения маршрутов на пересеченной местности.

Литература

1. М. де Берг, О. Чеонг, М. Кревельд, М. Овермарс «Вычислительная Геометрия» // ДМК Пресс, 2017
2. T. Sevilmiş, B. Mizrahi «Implementation of the visibility graph construction for a set of polygons for 2D in $O(n^2)$» // Bilkent University, 21p., 2021

3. H. Rohnert «Shortest paths in the plane with convex polygonal obstacles» // Inform. Process. Lett., pp.71–76, 1986
4. Timothy M. Chan «Optimal output-sensitive convex hull algorithms in two and three dimensions» // Discrete and Computational Geometry, Vol. 16, pp.361–368, 1996
5. F.P. Preparata «An Optimal Real-Time Algorithm for Planar Convex Hulls» // University of Illinois, 4 p., 1979
6. M. Shamos «Computational geometry» // Yale University, pp. 92-95, 1978
7. Botea A., Muller M., Schaeffer J. «Near Optimal Hierarchical Path-Finding (HPA*)» // Journal of Game Development, №3, pp.1-30, 2004
8. Urs Ramer, «An iterative procedure for the polygonal approximation of plane curves» // Computer Graphics and Image Processing, 1(3), pp.244–256, 1972

References

1. M. de Berg, O. Cheong, M. Kreveld, M. Overmars «Computational Geometry» // DMK Press, 2017
2. T. Sevilmiş, B. Mizrahi «Implementation of the visibility graph construction for a set of polygons for 2D in $O(n^2)$» // Bilkent University, 21p., 2021
3. H. Rohnert «Shortest paths in the plane with convex polygonal obstacles» // Inform. Process. Lett., pp.71–76, 1986.
4. Timothy M. Chan «Optimal output-sensitive convex hull algorithms in two and three dimensions» // Discrete and Computational Geometry, Vol. 16, pp.361–368, 1996.
5. F.P. Preparata «An Optimal Real-Time Algorithm for Planar Convex Hulls» // University of Illinois, 4 p., 1979
6. M. Shamos «Computational geometry» // Yale University, pp.92-95, 1978
7. Botea A., Muller M., Schaeffer J. «Near Optimal Hierarchical Path-Finding (HPA*)» // Journal of Game Development, N3, pp.1-30, 2004
8. Urs Ramer, «An iterative procedure for the polygonal approximation of plane curves» // Computer Graphics and Image Processing, 1(3), pp.244–256, 1972