О ВЫЧИСЛИТЕЛЬНЫХ АСПЕКТАХ РАСЧЕТА ПОЛЯ РАССТОЯНИЙ В ЗАДАЧЕ МОДЕЛИРОВАНИЯ ДВИЖЕНИЯ ЛЮДЕЙ

Попел Егор Викторович

Сибирский федеральный университет, Красноярск, Россия

e-mail: evpopel@gmail.com

Введение

Одним из актуальных направлений математического моделирования является создание математических моделей движения людей. Такие модели используются при решении задач комплексной безопасности в части оценивания времени эвакуации людей из помещений, зданий, сооружений при различных сценариях развития чрезвычайной ситуации.

Среди существующих моделей движения людей имеется многочисленный класс моделей, в которых главной “движущей силой” является минимизация расстояния до цели (выхода). Информация о расстоянии из каждой точки рассматриваемой плоской области до ближайшего выхода хранится, в так называемом, поле расстояний. Это поле является статическим и не зависит от положения людей.

Для вычисления поля вся область моделирования покрывается двумерной ортогональной сеткой с равными ребрами фиксированного размера, для каждого узла которой рассчитываются расстояния до выходов. Если узел сетки попал на непроходимый участок (такой как мебель, стена), то информация об этом также хранится в поле расстояний – расстояние до выхода будет равно бесконечно большому числу. В случае, когда потребуется вычислить расстояния в точке, не попавшей на узел сетки, будет использоваться билинейная интерполяция относительно четырех соседних к этой точке узлов. Шаг сетки определяет точность расчета поля и качество моделирования движения людей, поэтому наилучшим вариантом является мелкая сетка. Но тогда процедура расчета расстояний занимает много времени и требует немалых аппаратных ресурсов. В то же время рассчитывать поле расстояний необходимо каждый раз, когда вносятся какие-либо изменения в планировку здания, или даже в случае перестановки мебели. Таким образом, остро стоит задача создания такого алгоритма расчета расстояний, чтобы он выдавал корректные результаты как можно быстрее, и в то же время, чтобы результаты эти наименее отличались от истинных значений.

Вторая задача заключается в том, что полученную информацию о поле расстояний необходимо в каком-то виде хранить на жестком диске, чтобы в дальнейшем использовать ее для работы других модулей по моделированию движения людей. Область моделирования может иметь произвольную форму, и, при покрытии такой области непрерывной сеткой, площадь сеточного пространства оказывается больше площади области моделирования (в практических задачах речь идет о площадях, измеряемых десятками тысяч квадратных метров). В результате будет отводиться дисковое пространство под ненужную однообразную информацию. Вопрос об определении оптимального формата хранения поля расстояний также рассматривается в данной работе.

Постановка задачи

Границы поля (здания или этажа) заданы набором векторов (направленных отрезков) – прямых участков стен, векторы заданы координатами их начала и конца в

некоторой декартовой системе координат. Подразумевается, что вектора образуют замкнутую ломаную без самопересечений. Причем вектора ориентированы так, что внутренность здания лежит по правую сторону от вектора. По тому же принципу векторами заданы выходы из здания (или с этажа). Каждый из этих отрезков-выходов принадлежит одному из отрезков, задающих стены. Внутри полученного многоугольника таким же образом (множеством векторов, образующих замкнутые кривые без самопересечений) заданы элементы препятствий (мебель и т.п.).

Расчетной областью будем называть все пространство внутри здания за вычетом пространства, занимаемого препятствиями.

Требуется получить оценку расстояния в каждой точке поля до каждого из выходов. Расстояние в данном случае – длина минимальной траектории от точки поля до выхода, причем ни один из участков траектории не должен пересекать стены и препятствия. Оценка расстояния в каждой точке поля должна быть оптимальна, т.е. максимально удовлетворять каждому из противоречащих друг другу критериев:

1) минимальность погрешности оценки расстояния (минимальная разность между полученным значением расстояния и истинным, евклидовым значением длины траектории);
2) минимальность времени работы алгоритма расчета значения поля в каждой точке.

Вычислив требуемую оценку, необходимо организовать хранение полученных результатов – записать их в файл. Формат записи должен удовлетворять следующим параметрам:

1) объем полученного файла должен быть минимальным;
2) процедура последующего считывания поля расстояний из файла должна занимать как можно меньшее время.

Решение задачи

Способ расчета поля расстояний

Для расчета расстояния в каждой точке поля до выхода вся расчетная область покрывается достаточно мелкой квадратной сеткой, в узлах которой и будет рассчитываться расстояние. Для покрытия сеткой всей расчетной области на основании начальных данных находятся $x_{min}, y_{min}, x_{max}, y_{max}$ - минимальные и максимальные координаты концов отрезков, задающих положение стен. Эти координаты и будут задавать координаты крайних узлов сетки, которая будет иметь форму прямоугольника. Хранение в оперативной памяти информации о расстоянии для каждого узла логически будет осуществляться в виде прямоугольной матрицы, число элементов в которой равно количеству узлов сетки. Шаг сетки обозначим d. Тогда через порядковые номера строки и столбца для ячейки можно выразить ординату и абсциссу узла, соответствующего выбранной ячейке матрицы:

$$\begin{cases} x_{ij} = x_{min} + j \cdot d \\ y_{ij} = y_{max} - i \cdot d \end{cases}$$

Разумеется, для каждого выхода будет свое поле расстояний и, соответственно, своя матрица узловых значений сетки. Опишем алгоритм расчета расстояний для одного произвольного фиксированного выхода. При этом все другие выходы будем считать непроходимым препятствием. Поля расстояний для всех остальных выходов будут рассчитаны аналогично.

Итак, для начала заполним матрицу значений изначальными величинами. Если какой-то узел сетки попал на линию выхода, то, очевидно, расстояние для этого узла до выхода равно нулю. Это значение и заносим в матрицу. Узлов, попавших на линию выхода, может быть несколько (в таком случае для каждого из них в матрицу запишем нули), а может случиться так, что ни один узел сетки не попадет на линию выхода. В этой ситуации будем находить опорные узлы сетки по следующему принципу:

- по геометрическому расположению выхода найти узлы, ближайшие к линии выхода. Узлы ищем те, что находятся справа от вектора, задающего линию выхода, т.е. эти узлы лежат внутри помещения (расчетной области);

- сгустить сетку в районе линии выхода, то есть в отдельные переменные записать координаты пересечения ребер сетки и линии выхода - узлы дополнительной сетки;

- узлам основной сетки, ближайшим к линии выхода, присвоить евклидово расстояние до линии выхода: каждому узлу найти среди узлов доп.сетки узлы с равными координатами по одной из осей (ординате или абсциссе), в итоге, если линия выхода не параллельна координатным осям, наберется всего 3 узла (1 из основной сетки, 2 из доп.), которые являются вершинами прямоугольного треугольника с известными сторонами. Искомое значение узла основной сетки будет равно длине высоты такого треугольника, опущенной на гипотенузу.

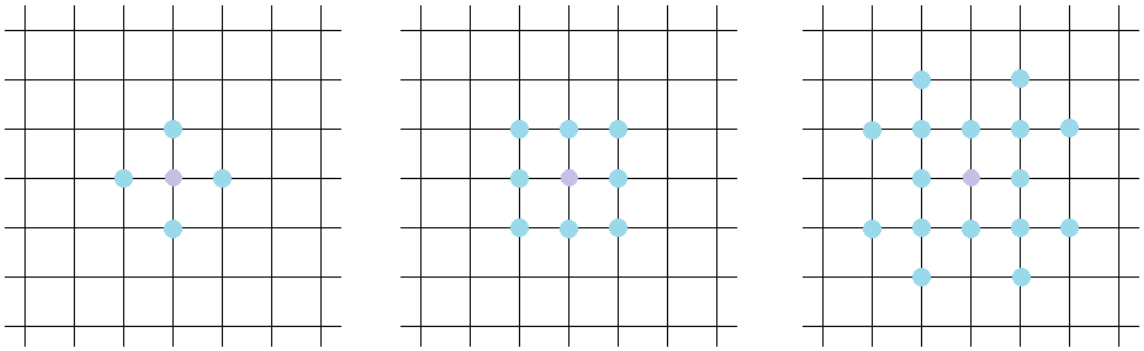
Теперь у нас имеются один или несколько узлов сетки, называемых «опорными», до которых мы будем искать расстояние. Все остальные ячейки матрицы заполняем максимальным значением (для типа double).

Алгоритм поиска расстояния между узлами сетки аналогичен алгоритму поиска минимального пути на графе, в таком случае в качестве вершин графа рассматриваются узлы сетки, которые соединяются между собой по определенному шаблону формирования смежных узлов.

Основными алгоритмами поиска минимального пути в графе являются: алгоритм Беллмана-Форда, Флойда-Варшалла, Дейкстры. Однако алгоритм Беллмана-Форда эффективен на разреженных графах, в то время как у нас у каждой вершины имеется порядка десяти смежных к ней вершин (какие узлы сетки мы будем считать смежными, мы рассмотрим дальше). Алгоритмы Дейкстры и Флойда-Варшалла выполняются за время одинакового порядка, но на практике после реализации каждого из алгоритмов было выявлено, что метод Дейкстры во всех случаях опережает действие алгоритма Флойда-Варшалла.

Далее – применяем алгоритм Дейкстры для расчета расстояний до опорных узлов от всех остальных узлов, попавших на расчетную область. Так как мы рассчитываем расстояние до нескольких вершин-узлов, то предварительно занесем в очередь все опорные узлы.

Для получения результатов с разной погрешностью можно использовать разные шаблоны смежных узлов.



а) б) в)

Рис. 1. Шаблоны смежных узлов
а) 4-узловой б) 8-узловой в) 16-узловой

В 4-узловом шаблоне расстояние до каждого соседнего (смежного) узла равно d. В 8-узловом шаблоне добавляются диагональные узлы, длина ребра до которых равна $d\sqrt{2}$. В 16-узловом шаблоне добавляются еще дальние узлы, расстояние до них от центрального узла: $d\sqrt{5}$. Также можно рассматривать еще более полные шаблоны смежных узлов.

Очевидно, чем большее число смежных узлов входит в шаблон, тем точнее вычисляется расстояние между узлами. Причем в сравнении с точным евклидовым расстоянием получаемое нами расстояние будет всегда больше.

Рассмотрим увеличение ошибки расстояния с уменьшением числа узлов в шаблоне на примере, рис. 2. Положим шаг сетки равным 1 см. Расстояния между помеченными узлами:

евклидово: 18,47 см
16-узловой: 19,60 см
8-узловой: 20,12 см
4-узловой: 26,00 см

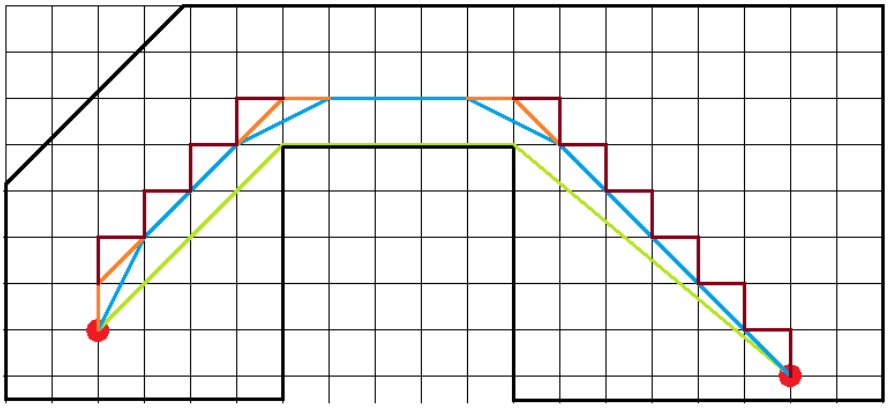


Рис. 2. Пример расстояний между двумя узлами
Черный цвет – стены, зеленый цвет – истинное расстояние, голубой – 16-узловой шаблон, оранжевый – 8-узловой шаблон, бардовый – 4-узловой шаблон

В рассмотренном примере очень мало промежуточных узлов между двумя узлами, расстояние между которыми измерялось, поэтому видно очень резкое возрастание погрешности. В более масштабных примерах общая тенденция сохраняется:

евклидово: 100%

16-узловой: ~104%
8-узловой: ~106-107%
4-узловой: >110%

Помимо других вопросов для реализации алгоритма Дейкстры необходимо на каждом этапе (рассмотрении очередного узла) исключать из рассмотрения те узлы, с которыми смежен данный узел согласно шаблону, но соединить их напрямую нельзя из-за препятствия. Для этого при рассмотрении очередной пары смежных узлов будем проводить проверку на пересечение ребром какого-либо отрезка, определяющего стену или препятствие. Таким образом, требуется провести проверку на пересечение двух отрезков.

Вместо пересечения отрезков выполним проверку на пересечение двух прямых, задаваемых этими отрезками. В результате, если прямые не параллельны, получим какую-то точку, которую надо проверить на принадлежность обоим отрезкам. Для этого достаточно проверить, что эта точка принадлежит обоим отрезкам в проекции на ось X и ось Y.

Если же прямые оказались параллельными, то, если они не совпадают, то отрезки точно не пересекаются. Если же прямые совпали, то отрезки лежат на одной прямой, и для проверки их пересечения достаточно проверить, что пересекаются их проекции на оси X и Y.

Способ хранения поля расстояний

Для хранения информации о поле расстояний рассчитанные расстояния для узлов сетки записываются в файл. В файл записывается несколько матриц одинаковой размерности - для каждого выхода своя матрица, каждая ячейка которой содержит число типа double (дробное число, в памяти компьютера занимает 64 бита) - расстояние до одного из выходов. Порядковые номера строчки и столбца для ячейки - это относительные целочисленные координаты узла сетки. Также в файле хранится шаг сетки d и сдвиг сетки относительно начала общей системы координат x_{min} и y_{max}. Только тогда у нас есть полная информация о расстоянии до выхода для каждой координаты в изначально установленной системе координат. Таким образом, в файле записаны только огромные массивы дробных чисел - расстояний.

Однако немалая часть узлов сетки в виде прямоугольника приходится не на расчетную область, и после расчета в этих ячейках лежат одни и те же числа – те самые максимумы, которые мы записали в ячейки матрицы в самом начале работы алгоритма. По сути – это ненужная нам информация, т.к. нам достаточно знать значения поля лишь в пределах расчетной области. А процент таких “ненужных” узлов в некоторых зданиях может достигать 50% и даже больше (например, популярные для ВУЗов корпуса в виде колодцев: на верхних этажах на месте скважины – непроходимое “препятствие”). Поэтому я разработал способы для записи в файл информации о поле расстояний без фиксирования “ненужных” узлов.

Первый способ заключается в следующем: мы знаем, что в файле записаны только неотрицательные дробные числа (неотрицательные - т.к. это расстояния). Формат записи дробных чисел: 64 бита, причем первый бит определяет знак числа (0 - плюс, 1 - минус), далее 63 бита под экспоненту и мантиссу числа. Таким образом, мы знаем, что все числа в файле начинаются с бита "0". Теперь между всеми числами (между каждыми двумя)

вставим по одному биту "1". Затем в каждой строке матрицы удалим расстояния для ненужных узлов. Тогда на месте этих значений останутся подряд идущие разделительные биты "1". И при считывании файла мы будем идти по строкам так: считали один бит "1", затем смотрим: следующий бит - "0"? Если да, то считываем следующие 63 бита – само число, увеличиваем индекс в строке на 1. Если после разделительного бита - стоит "1", то значит это непроходимый узел, записываем в ячейку максимальное число и увеличиваем индекс на 1.

Весь способ кратко представлен на рис. 3.

$$\dots\ R_{i,j}\boxed{1}R_{i,j+1}\boxed{1}R_{i,j+2}\boxed{1}R_{i,j+3}\ \dots$$

$$\boxed{1}0\underbrace{1001\ \dots\ 1101}_{63\text{ бита}}\boxed{1}\boxed{1}0\underbrace{0111\ \dots\ 1001}_{63\text{ бита}}\boxed{1}$$

$$R_{i,j} \qquad\qquad R_{i,j+2}$$

Рис. 3. Наглядное представление способа формирования файла

Итого: мы вставили рядом с каждым 64-битным числом по одному биту - объем файла увеличился на 1/64. Но при этом убрали все ненужные узлы. Объем файла сократился на количество этих узлов. Если их 50% - то файл уменьшится в два раза.

Другой метод основан на заключении, что так как для многих узлов сетки расстояния до выхода одинаковые (для радиально удаленных от линии выхода), то можно записать в файл следующую структуру. Каждая строка - для своего уникального значения расстояния до выхода. Формат строки: само значение расстояния, а затем перечисление координат (целочисленных) узлов, для которых это расстояние является расстоянием до выхода. Под запись целых чисел в памяти отводится 32 бита, то есть под запись двух координат (X и Y) нужны те же самые 64 бита. В итоге, в файл также не будет записана ненужная информация, однако файл, созданный данным способом, будет даже несколько больше объемом, чем при использовании предыдущего способа, потому что значения расстояний займут больше, чем 1/64 объема файла. Еще один недостаток данного метода в сравнении с предыдущим: процедура считывания этого файла будет еще сложнее, т.к. необходимо запоминать значения расстояний и считывать в два раза больше чисел – две координаты.

Таким образом, самым выгодным из трех форматов файла является второй, с разделительными битами, потому что он значительно выигрывает в объеме в сравнении с простой записью матрицы (причем линейно – чем больше ненужных узлов, тем на такую же величину меньше файл), и незначительно превосходит третий способ по обоим критериям – объему файла и скорости его чтения.

Итого, найдено два пути решения второй задачи, и создание такого формата записи в файл, чтобы ещё сильнее сократить его объем, не представляется возможным.

ЛИТЕРАТУРА

1. Schadschneider A., Klingsch W., Kluepfel H., Kretz T., Rogsch C., Seyfried A. Evacuation Dynamics: Empirical Results, Modeling and Applications // Encyclopedia of Complexity and System Science, Springer, 2009, p. 3142-3197.