Полученные результаты говорят о том, что модель построенная на основе данных Uber Movement не отвечает требованиям адекватности по GEH метрике, однако регрессионный анализ показывает причину таких значений. Из графика видно, что моделируемый поток в среднем на 17% ниже реальных данных. Возможными путями исправления модели являются:

- введение дополнительных коэффициентов в формулу (1)
- калибровка имитационной модели на данных детекторов.

Литература

1. Михайлов А.Ю., Головных И.М., Современные тенденции проектирования и реконструкции улично-дорожных сетей городов// – Новосибирск: Наука, – 2004. – с.267

2. Швецов В. И., Математическое моделирование транспортных потоков// Автомат. и телемех., – 2003. – №11. – с.3–46.

3. Lopez P.A. et al., Microscopic Traffic Simulation using SUMO, 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 2018, pp. 2575-2582, doi: 10.1109/ITSC.2018.8569938

4. Feldman O., The GEH Measure And Quality Of The Highway Assignment Models, European Transport Conference At: Glasgow, October 2012

5. Шашков, В.Б. Прикладной регрессионный анализ. Многофакторная регрессия// Учебное пособие. – Оренбург: ГОУ ВПО ОГУ, – 2003. – с.363.

А.А. Мещеряков

АЛГОРИТМ ПОСТРОЕНИЯ ОПТИМАЛЬНОГО МАРШРУТА ДЛЯ МОРСКИХ СУДОВ

(Самарский университет)

Одной из важных проблем судоходства является задача построения оптимального маршрута с учетом метеорологических условий и обеспечения безопасности прохождения пути. Построение оптимального по времени маршрута с учетом данных прогноза погоды особенно важно с точки зрения сокращения экономических затрат. Поэтому проведение исследований по оптимизации маршрута по указанным критериям крайне актуально в навигации.

Несмотря на быстрое развитие информационных технологий в последние годы, до сих пор не существует универсального общепринятого метода построения оптимальных маршрутов, гарантирующего вычислительную простоту и точность прогноза.

Для решения задачи построения оптимального маршрута сначала необходимо описать математическую модель. Скорость судна при плавании в данной модели описывается следующим соотношением:

$$V = \mathrm{V}_0 + \mathrm{V}_{\mathrm{wt}}\alpha + \mathrm{V}_{\mathrm{wd}}\beta,$$

Где V_0 - собственная скорость судна;

V_{wt} - скорость течения;
V_{wd} - скорость ветра;
α - коэффициент угла между курсом и направлением течения;
β - коэффициент угла между курсом и направлением ветра;

Длительность движения судна будет изменяться в зависимости от степени воздействия внешних факторов на отдельных отрезках маршрута. В общем случае

$$t = \frac{S_1}{V_1} + \frac{S_2}{V_2} + \ldots + \frac{S_n}{V_n},$$

Где $S_1, S_2, \ldots S_n$— отрезки пути судна, на которых наблюдаются разные гидрометеорологические условия, мили;
$V_1, V_2, \ldots V_n$— абсолютная скорость судна на соответствующих участках пути, узлы;

Таким образом, общее время плавания с учетом скорости и направления ветра и течения на всех промежутках пути выражается следующей формулой:

$$t = \frac{S_1}{V_0 + V_{1wt}\alpha_1 + V_{wd1}\beta_1 + t_{ST}} + \frac{S_2}{V_0 + V_{2wt}\alpha_2 + V_{wd2}\beta_2} + \ldots + \frac{S_n}{V_0 + V_{nwt}\alpha_n + V_{wdn}\beta_n}$$

Из формулы следует, что мы должны решить следующую задачу оптимизации для каждого участка пути S:

$$\begin{cases} S \to \min \\ V_{wt}\alpha \to \max \\ V_{wd}\beta \to \max \end{cases}$$

Разбиение на участки S будет проводиться в тех точках, где меняется направление ветра или течения, либо в следствие обхода препятствий меняется курс судна.

Данная математическая модель обладает аддитивностью и оптимальной подструктурой, что позволяет решать рассматриваемую задачу в терминах динамического программирования.

В основе метода лежит идея рассмотрения исходной задачи как представителя семейства сходных с ней задач. Данный метод применим к задачам с оптимальной подструктурой, выглядящим как набор перекрывающихся подзадач, сложность которых чуть меньше исходной. Подход динамического программирования состоит в том, чтобы решить каждую подзадачу только один раз, сократив тем самым и количество вычислений.

Оптимальная подструктура в динамическом программировании означает, что оптимальное решение подзадач меньшего размера может быть использовано для решения исходной задачи. В общем случае мы можем решить задачу, в которой присутствует оптимальная подструктура, проделывая следующие три шага.

1. Разбиение задачи на подзадачи меньшего размера.
2. Нахождение оптимального решения подзадач рекурсивно, проделывая такой же трехшаговый алгоритм.

3. Использование полученного решения подзадач для конструирования решения исходной задачи.

Одним из основных свойств задач, решаемых с помощью динамического программирования, является аддитивность. Неаддитивные задачи решаются другими методами.

Одним из классических примеров методов динамического программирования является алгоритм Дейкстры. Данный алгоритм находит кратчайшие пути от одной из вершин графа до всех остальных.

Сложность алгоритма Дейкстры зависит от способа нахождения вершины, расстояние до которой ищется, а также способа хранения множества непосещённых вершин и способа обновления меток. Обозначим через n количество вершин, а через m — количество рёбер в графе G.

В простейшем случае, когда для поиска вершины с минимальным расстоянием d[v] просматривается всё множество вершин, а для хранения величин d используется массив, время работы алгоритма есть $O(n^2)$. Основной цикл выполняется порядка n раз, в каждом из них на нахождение минимума тратится порядка n операций. На циклы по соседям каждой посещаемой вершины тратится количество операций, пропорциональное количеству рёбер m (поскольку каждое ребро встречается в этих циклах ровно дважды и требует константное число операций). Таким образом, общее время работы алгоритма $O(n^2 + m)$, но, так как $m \leq n(n-1)$, оно составляет $O(n^2)$.

Данная сложность алгоритма является существенным недостатком при большом количестве узлов. В случае решения задачи поиска оптимального маршрута для морских судов, вершинами графа являются географические координаты с подходящей для данного судна глубиной. Точность построения маршрута (и, как следствие, безопасность прохождения пути) при этом растет с увеличением масштаба координат, что в свою очередь приводит к росту количества вершин графа и значительному увеличению времени работы алгоритма.

Исходя из этой особенности, алгоритм Дейкстры плохо применим для построения длинных маршрутов в открытых водах.

Популярным улучшением алгоритма Дейкстры является алгоритм A*, который представляет собой поиск первого наилучшего совпадения на графе, который так же учитывает стоимость перехода от одной вершины к другой.

Порядок обхода вершин определяется эвристической функцией:

$$f(x) = h(x) + g(x),$$

Где $h(x)$ - функция оценки расстояния от текущей вершины до конечной;
$g(x)$ - функция стоимости достижения текущей вершины из начальной;

Сложность данного алгоритма оценивается следующим образом:

$$|h(x) - h^*(x)| \leq O(\log h^*(x)),$$

Где h^* - оптимальная эвристика, то есть точная оценка расстояния из текущей вершины x до конечной;

Данный алгоритм работает существенно быстрее, чем алгоритм Дейкстры, однако он не способен корректировать уже пройденную часть марш-

рута в случае, если на пути встречается препятствие и его можно обойти более оптимальным способом.

Для устранения вышеназванных недостатков рассмотренных алгоритмов был разработан новый алгоритм, основанный на методе A*, учитывающий препятствия на пути и обходящий их оптимальным способом. Для целей данной работы назовем разработанный алгоритм «модифицированный A*».

Модифицированный алгоритм A* рекурсивно использует принципы динамического программирования. Путь из начальной в конечную точку разбивается на участки, в которых отсутствуют препятствия и курс судна остается неизменным. На граничных точках между участками, где необходимо обойти препятствие прежде, чем попасть на следующий прямолинейный участок, используется метод обхода вершин графа из алгоритма A*.

Псевдокод разработанного алгоритма:

Обозначения

- Z[1..n][1..k] - матрица координат, в которой точки с подходящей для судна глубиной обозначены числом 0, мелководье - числом 1.
- a — текущая начальная вершина
- b - текущая точка
- A - начальная вершина, от которой строится маршрут
- B - конечная вершина, до которой строится маршрут
- U — множество посещённых граничных вершин
- L - множество непосещениях граничных точек препятствия
- d(A,B) — по окончании работы алгоритма равно длине кратчайшего пути из A до вершины B

Присвоим $a \leftarrow A, d(A, B) = 0$

Пока Z[i][k] == 0 или $b \neq B$, строим прямолинейный маршрут от a до B, проходящий через координаты Z[i][k].

Если Z[i][k] == 1, добавляем точку b с координатами Z[i][k] в L.

Пока L не пусто:

Сортируем L по расстоянию до конечной точки B

Если путь из a в b является касательным к текущему препятствию:

d(A,B) += d(a,b)

Запускаем алгоритм с параметрами a = b

Иначе:

добавляем все соседние вершины от b в L, если они тоже граничные точки препятствия.

Удаляем b из L

Добавляем b в U

Данный алгоритм позволяет избежать излишней кривизны пути при обходе препятствий за счет повторного пересчета маршрута на некоторых участках для обхода препятствия по касательной.

Литература

1. Левитин А. В. Глава 9. Жадные методы: Алгоритм Дейкстры // Алгоритмы. Введение в разработку и анализ — М.: Вильямс, 2016. — С. 189—195. — 576 с. — ISBN 978-5-8459-0987-9

2. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ = Introduction to Algorithms. — 2-е изд. — М.: «Вильямс», 2014. — С. 1296. — ISBN 0-07-013151-1.

3. Рассел, Стюарт (2018). Искусственный интеллект: современные методы. (4-ое издание.). Бостон: Пирсон. ISBN 978-0134610993. OCLC 1021874142.

Т.И. Михеева, Н.М. Клепиков, С.В. Михеев

МЕТОДЫ РАСПОЗНАВАНИЯ ДОРОЖНЫХ ЗНАКОВ В СРЕДЕ «ITSGIS»

(Самарский университет)

Методы распознавания дорожных знаков, используемые в интеллектуальной транспортной геоинформационной системе «ITSGIS», реализуются на использовании аппаратных и программных модулей: камера, плата захвата изображения, база данных, модуль обнаружения, модуль классификации.

Изображение с камеры поступает на вход системы. Затем алгоритмом определяется дислокация дорожного знака. После этого знак распознается плагином распознавания.

База данных содержит схему в зависимости от поставленных перед распознаванием задачи.

Системы могут распознавать следующие знаки дорожного движения:

- ограничение скорости;
- въезд запрещен;
- обгон запрещен;
- отмена запрета обгона;
- движение прямо;
- движение налево;
- движение направо;
- движение прямо или налево;
- движение прямо или направо;
- объезд препятствия слева;
- объезд препятствия справа;
- переходный переход;
- главная дорога;
- уступите дорогу.

Точность распознавания заметно уменьшается, если знак дорожного движения испачкан, частично перекрыт чем-либо или наклонен.