**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**Кафедра дискретной математики и алгоритмики**

КАСИЯНИК

Алексей Леонидович

**ОТЧЕТ №2 О РАБОТЕ В РАМКАХ МАГИСТЕРСКОЙ ДИССЕРТАЦИИ**

специальность 1-31 81 09 «Алгоритмы и системы обработки больших объемов информации»

Минск, 2016

Оглавление

ПОСТАНОВКА ЗАДАЧИ 3

ПЕРЕЧЕНЬ ВОПРОСОВ 5

ОБЗОР ТЕОРИИ О МНОГОКРИТЕРИАЛЬНОЙ ЗАДАЧЕ ОПТИМИЗАЦИИ И МЕТОДОВ ЕЕ РЕШЕНИЯ 6

1.1 Постановка задачи 8

1.2 Методы решения 9

1.2.1 Лексикографическое упорядочение критериев 9

1.2.2 Метод последовательных уступок по значению ведущего критерия 9

2.2.3 Метод главного критерия 11

2.2.5 Свертка критериев 11

ПЛАН ДАЛЬНЕЙШИХ ИССЛЕДОВАНИЙ 14

# ПОСТАНОВКА ЗАДАЧИ

Автоматическое создание городских туристических маршрутов является интересной и востребованной задачей, которая приобретает особенную важность в контексте современных требований и вызовов. Современный пользователь имеет широкий выбор решений, предлагающий сервис планирования путешествий и маршрутов, однако большинство таких приложений имеют весьма ограниченный функционал: например, учитывают (или не учитывают вовсе) только какой-то конкретный вид транспорта. Но особенно важно то, что существующие решения практически практически полностью игнорируют ключевой функционал – многокритериальную оптимизацию.

Задача многокритериального многомодального построения туристических маршрутов особенно интересна в контексте стремительно растущей проблемы трафика на дорогах, имеющей значительное влияние на экономику и окружающую среду. Операторы трафика рассматривают использование многомодального транспорта (общественный транспорт, велосипед, park'n'ride и т.д.) в качестве хорошей стратегии решения проблем перегруженности и пагубного воздействие на окружающую среду.

Многомодальный многокритериальный планировщик путешествий предоставляет пользователю различные многомодольные варианты, оптимизированные по критериям, которые предпочитает пользователь (общая стоимость путешествия, скорость и время, количество мест для посещения, экологичность).

Построение многомодальных и многокритериальных маршрутов само по себе является сложной математической задачей, которая существенно усложняется необходимостью обработки и интеграции больших объемов данных широкого спектра, в том числе информацию о дорожной сети, точках интереса (POI), расписания общественного транспорта, а также необходимостью оптимизации для конкурирующих критериев, где полная оптимизация для одного критерия, например, время в пути, может отрицательно повлиять на другие критерии, например, стоимость и экологичность (выброс CO2 в атмосферу). Взаимосвязь между такими критериями может быть иметь весьма субъективный характер, так как их приоритеты будут варьироваться в зависимости от пользователя к пользователю.

В рамках данной работы ставится задача разработки алгоритма создания многомодальных многокритериальных маршрутов для городского туризма.

ПЕРЕЧЕНЬ ВОПРОСОВ

* Изучение литературы по тематике магистерской диссертации «Многокритериальная многомодальная задача планирования туристического машрута».
* Анализ существующих подходов к решению поставленной в магистерской диссертации задачи.

ПОСТРОЕНИЕ МОДЕЛИ

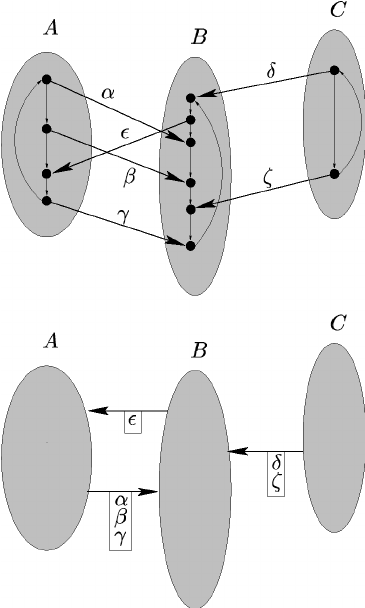
2.1 «Развернутый во времени» и «изменяющийся во времени» графы

Существует два подхода к моделированию пространственных данных с учетом зависимости от времени:

* построение «развернутого во времени» графа (time-expanded approach)
* построение «изменяющегося во времени» графа (time-dependent approach).

При построении «развернутого во времени» графа каждая вершина рассматривается как пара (l, t), где l – географическая локация, t – время события. Событиями являются все отправления и прибытия транспорта в данную локацию. Например, в данной модели все отправления поездов с вокзала будут рассматриваться как самостоятельные вершины в графе.

При построении «изменяющегося во времени» графе вершины не ассоциируются с временем и будут представлять только локацию.

Рис. 1 “Развернутый во времени” граф (сверху) и “изменяющийся во времени” граф

2.2 Моделирование с помощью «изменяющегося во времени» графа

Обозначим через множество географических точек, которые включают точки интереса (POI – point of interest), вокзалы, остановки общественного транспорта с фиксированным расписанием и места, в которых турист может воспользоваться транспортом без фиксированного расписания (например, парковки такси).

Построим модель взвешенной сети используя «изменяющийся во времени» граф , где – множество вершин, – множество ребер. Вершины представлены в виде пар где – географическая локация, – время события. Две вершины и соединены ребром тогда и только тогда, если турист может отправиться из точки в момент времени и прибыть в точку в момент времени , используя транспортный режим (вид транспорта) . Таким образом, если несколько видов транспорта позволяют выехать из точки в момент и прибыть в в момент , то вершины будут соединены несколькими параллельными дугами, отличающимися в .

Разрабатываемая модель также должна предусматривать ситуацию, когда турист должен задержаться в каком-то месте на определенное время. Например, если локация является музеем, то возможно туристу придется подождать открытия. В случае, если точка является вокзалом, то модель должна описывать время ожидания пересадки. Для этого введем дополнительный транспортный режим – *режим ожидания*. В таком случае будем полагать, что вершины и соединены дугой ожидания

Каждая дуга будет иметь множество весов:

* – расстояние между вершинами, которое будет пройдено используя режим
* – средняя скорость для режима на данном участке. Для транспорта с фиксированным расписанием будем полагать равной рекомендованной скорости. Для пешего и велосипедного режима , где – стандартная средняя скорость, – высокая средняя скорость.
* – cтоимость путешествия при использовании режима на данном участке ( – для пешего и велосипедного режима и режима ожидания).
* – количество выбрасываемого CO2 на человека ( – для пешего, велосипедного режима и режима ожидания).

2.3 Формулирование задач булевского программирования

В данной главе опишем формулирование задач булевского программирования для модели описанной в предыдущей главе. Предположим, что пользователь выбрал все возможные критерии и упорядочил их следующим образом:

1. максимизация количества посещенных точек интереса (POI),
2. минимизация времени прибытия в конечную точку маршрута,
3. минимизация стоимости путешествия
4. минимизация количества изменений транспортных модулей,
5. минимизация общего времени пересадок и ожидания,
6. минимизация общего времени в пешем модуль,
7. минимизация выброса .

Составление задач для иного порядка критериев происходит по аналогичной схеме.

2.3.1 Максимизация количества посещенных точек интереса

Задача булевского программирования для максимизации количества посещенных точек формулируется следующим образом:

где

– множество точек POI, выбранных пользователем для посещения,

V – множество дуг посещения.

Сформулируем условия для данной задачи.

Маршрут должен содержать в точности одну дугу, выходящую из начальной вершины :

Маршрут должен содержать в точности одну дугу, входящую в конечную вершину :

Нужно гарантировать, что все промежуточные вершины пути посещаются единожды. Для каждой точки маршрута не может быть больше одной входящей и выходящей дуги:

Количество входящих и выходящих дуг из промежуточных вершин маршрута должно быть равно:

Не более одной дуги пересадки для каждой вершины пути:

Каждая точка интереса посещается не более одного раза:

В начальный момент времени дуги выходят только из стартовой точки :

Переключение между модулями происходит только посредством перехода по трансферной дуге:

2.3.2 Минимизация времени прибытия в конечную точку маршрута

2.3.3 Минимизация стоимости путешествия

2.3.4 Минимизация количества изменений транспортных модулей

2.3.5 Минимизация общего времени пересадок и ожидания

2.3.4 Минимизация общего времени в пешем модуль

2.3.4 Минимизация выброса

ОБЗОР ТЕОРИИ О МНОГОКРИТЕРИАЛЬНОЙ ЗАДАЧЕ ОПТИМИЗАЦИИ И МЕТОДОВ ЕЕ РЕШЕНИЯ

2.1 Постановка задачи

Многокритериальная задача оптимизации может быть записана в следующем виде:

|  |  |
| --- | --- |
| , | (1.1.1) |

где - целевые функции, которые могут конфликтовать друг с другом, и - область допустимых решений.



Задачи максимизации можно привести к задачам минимизации следующим образом:

|  |  |
| --- | --- |
| , | (1.1.2) |

Цель многокритериальной задачи оптимизации – одновременно минимизировать все целевые функции.

Обозначим за отображение пространства решений в область допустимых целевых значений, т.е. каждому элементу пространства решений сопоставим вектор из значений целевых функций на этом элементе:



|  |  |
| --- | --- |
|  | (1.1.3) |

Так как мы предполагаем, что целевые функции могут конфликтовать друг с другом и не являются независимыми, то может не существовать единственного решения, которое сможет оптимизировать все целевые функции одновременно. Поэтому под оптимальным решением будем понимать решение, в котором невозможно улучшить значение одной целевой функции без ухудшения другой целевой функции.

2.2 Методы решения

2.2.1 Лексикографическое упорядочение критериев

Рассмотрим схему компромисса – лексикографическое упорядочение критериев (ЛУК). Данная схема предусматривает, что последовательность, в которой критерии перечислены, определяет их значимость в следующем смысле: каждый предшествующий критерий несравненно важнее любого из перечисляемых за ним. Синтез решения, оптимального при лексикографическом упорядочении критериев, реализуется следующим образом.

Сначала решаем однокритериальную задачу

|  |  |
| --- | --- |
|  | (1.2.4) |

Обозначим *D1* множество ее оптимальных решений. Если *D1* – одноэлементное множество, то единственное оптимальное решение задачи 12.4) является одновременно оптимальным по принципу ЛУК исходной многокритериальной задачи (1.1.1). В противном случае далее решаем задачу

|  |  |
| --- | --- |
|  | (1.2.5) |

Обозначим D2 множество оптимальных решений задачи (1.2.5). Если *D2* – одноэлементное множество, то единственное оптимальное решение задачи (1.2.5) является одновременно оптимальным по принципу лексикографического упорядочения критериев решением исходной многокритериальной задачи (1.1.1). В противном случае поступаем аналогично предыдущему – решаем задачу минимизации значения критерия *Q3(x)* при условии, что *х ∈ D2*, и т.д. Максимальное число последовательно решаемых однокритериальных задач равно m, т.е. числу критериев исходной многокритериальной задачи. Если решение задачи (1.1.1) определилось в результате выполнения меньшего числа этапов, мы остановились на *k*-ой итерации, то оно единственно. В противном случае может оказаться, что оптимальных (при имеющемся лексикографическом упорядочении критериев) решений этой задачи более чем одно; все они эквивалентны.

2.2.2 Метод последовательных уступок по значению ведущего критерия

Суть метода последовательных уступок по значению ведущего критерия поясним сначала на примере бикритериальной задачи максимизации

|  |  |
| --- | --- |
|  | (1.2.6) |

в которой критерий *Q1(x)* считаем ведущим. Реализуя этот метод, сначала находим решение *х\*,* оптимальное при лексикографическом упорядочении критериев *(Q1(x), Q2(x)).* Пусть *(Q1(x\*), Q2(x\*)) = (a, b).* Точка *(a, b)* – оценка найденного решения. Если данная оценка (первая ее координата *a* – максимально возможное значение критерия Q1(x), а вторая координата *b* – максимально возможное значение критерия *Q2(x)*) удовлетворяет лицо, принимающее решение, то *х\** принимается за искомое оптимально-компромиссное решение; в противном случае ЛПР назначает уступку *δ1, δ1 > 0* по допустимому значению первого критерия. Далее решается задача

|  |  |
| --- | --- |
|  | (1.2.7) |

при дополнительном условии

|  |  |
| --- | --- |
| *Q*1(*x*) ≥ *a* – δ1*.* | (1.2.8) |

Пусть оптимальное решение х\*\* задачи в исходной задаче (1.2.6) имеет оценку *(a1, b1).* Очевидно, *b1 ≥ b*. Если данная оценка удовлетворяет ЛПР, то *х\*\** принимается за искомое оптимально-компромиссное решение задачи (1.2.6) в противном случае ЛПР назначает новую уступку *δ2*, где *δ2 > δ1*, по допустимому значению первого критерия. Далее решается задача

|  |  |
| --- | --- |
|  | (1.2.9) |

при дополнительном условии

|  |  |
| --- | --- |
| *Q1(x) ≥ a – δ2.* | (1.2.10) |

Пусть оптимальное решение *х\*\*\** задачи (1.2.9) - (1.2.10) в исходной задаче (1.2.6) имеет оценку *(a2, b2).* Как очевидно, *b2 ≥ b1*. Если данная оценка удовлетворяет ЛПР, то *х\*\*\** принимается за искомое оптимально-компромиссное решение задачи (1.2.6); в противном случае ЛПР назначает новую уступку *δ3, δ3 > δ2*, по допустимому значению первого критерия, и т.д.

Описанный процесс продолжается вплоть до отыскания решения с оценкой, устраивающей лицо, принимающее решение, или вплоть до ситуации, когда дальнейшие уступки по допустимому значению первого критерия становятся невозможными.

2.2.3 Метод главного критерия

Метод главного критерия заключается в оптимизации значения наиболее важного критерия при условии, что остальные критерии принимают значения, не больше предписанных пороговых величин. Вводим нумерацию, при которой - главный критерий. Тогда задача (1.1.1) сводится к однокритериальной задаче



|  |  |
| --- | --- |
|  | (2.2.11) |

при дополнительных условиях

|  |  |
| --- | --- |
| , *k* = 2, 3, …, *m*; | (2.2.12) |

где *hk* – заданные соответственно для второго, третьего, …, m-го критериев пороги.

2.2.4 Свертка критериев

Метод линейной свертки позволяет перейти от многокритериальной задачи к однокритериальной задаче оптимизации. Целевая функция в последней является суммой целевых функций *Qi*, домноженных на весовые коэффициенты. Эти коэффициенты могут быть нормализованы, чтобы их сумма давала единицу, что не обязательно в общем случае.

Задачу линейной свертки критериев можно записать следующим образом

|  |  |
| --- | --- |
|  | (2.2.13) |

при условии х ∈ D.

При этом коэффициенты имеют следующие ограничения:

|  |  |
| --- | --- |
| , , | (2.2.14) |
|  | (2.2.15) |

Так как целевые функции могут иметь различную степень важности, а также иметь различные промежутки значений, это значит, что нам может понадобиться нормализация целевых функций для получения оптимального по Парето решения. В качестве нормализирующей схемы можно использовать нормализацию по разнице значений целевых функций в точках Надир (N) и Утопия (U), которая даст нам длину интервала, где оптимальное значение целевой функции может варьироваться в пределах Парето-фронта.

Можно заметить, что размеры множества Парето-оптимальных альтернатив предоставляют информацию для решения задачи. Компоненты идеального целевого вектора определяются минимизацией каждой целевой функции независимо от остальных, т. е.:



Идеальный вектор называемый точкой Утопия, не всегда может являться допустимым, так как могут существовать конфликты между целевыми функциями. Данная точка дает нам информацию о нижней границе Парето-фронта.



Верхние границы Парето-фронта определяются компонентами точки Надира:



Так как мы нормализируем целевые функции по интервалам их изменения в Парето-фронте, то для улучшения результатов нормализации мы будем использовать следующие коэффициенты:



Можно заметить, что после нормализации целевые функции будут ограничены:



что дает одинаковый интервал для изменения целевых функций.

Для вычисления нормализационного интервала необходимо решить k задач оптимизации вида для получения x[i]. Значение этих элементов является ключевым, так как они необходимы для вычисления и .



ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

3.1 Входные параметры

Планировщик туристических маршрутов предполагается реализовать в виде веб-сервиса. Пользовательский интерфейс должен предоставлять возможность задания критериев желаемого маршрута.

Обязательными начальными параметрами являются:

1. область построения маршрута, содержащая множество географических точек ,
2. - точка отправления,
3. - точка прибытия,
4. - наиболее раннее допустимое время отправления,
5. - наиболее позднее допустимое время прибытия.

При планировке маршрута пользователь также может задать следующие параметры:

1. – множество точек интереса, которые предполагается посетить,
2. – длительность посещения точек,
3. – множество допустимых транспортных модулей,
4. – средняя скорость перемещения пешком,
5. – средняя скорость перемещения на велосипеде,

Средняя скорость перемещения пешком и на велосипеде задается только в случае выбора соответствующих транспортных модулей пользователем.

Пользователь также будет иметь возможность оптимизировать маршрут по следующим критериям, выбирая и ранжируя их по степени важности:

1. максимизация количества посещенных точек интереса (POI),
2. минимизация времени прибытия в конечную точку маршрута,
3. минимизация стоимости путешествия
4. минимизация количества изменений транспортных модулей,
5. минимизация общего времени пересадок и ожидания,
6. минимизация общего времени в пешем модуль,
7. минимизация выброса .

**СРЕДСТВА РАЗРАБОТКИ**

Для анализа исходных данных, а также разработки последующих моделей планируется использование следующих технологий и инструментов обработки больших объемов информации:

* Apache Spark (в том числе GraphX)
* Язык программирования Java 1.8

*Apache Spark*— программный каркас с открытым исходным кодом для реализации распределённой обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов Hadoop. В отличие от классического обработчика из ядра Hadoop, реализующего двухуровневую концепцию MapReduce с дисковым хранилищем, использует специализированные примитивы для рекурентной обработки в оперативной памяти, благодаря чему позволяет получать значительный выигрыш в скорости работы для некоторых классов задач, в частности, возможность многократного доступа к загруженным в память пользовательским данным делает библиотеку привлекательной для алгоритмов машинного обучения.

Проект предоставляет программные интерфейсы для языков Java, Scala, Python, R. Состоит из ядра и нескольких расширений, таких как Spark SQL (позволяет выполнять SQL-запросы над данными), Spark Streaming (надстройка для обработки потоковых данных), Spark MLlib (набор библиотек машинного обучения), GraphX (предназначено для распределённой обработки графов). Может работать как в среде кластера Hadoop под управлением YARN, так и без компонентов ядра Hadoop, поддерживает несколько распределённых систем хранения — HDFS, OpenStack Swift, NoSQL-СУБД Cassandra, Amazon S3.

ПЛАН ДАЛЬНЕЙШИХ ИССЛЕДОВАНИЙ

При работе над магистерской диссертацией планируется продолжение работы на теоретической частью, изучение специфики входных данных и способов их обработки.

Также будут дополнительно изучены средства разработки, а также с их помощью будет создан прототип планировщика туристических маршрутов.