Оглавление

1. Введение

Основной целью ознакомительной практики 4-го семестра, входящей в учебный план подготовки бакалавров по направлению 01.03.04 — Прикладная математика, яв ляется знакомство с особенностями осуществления деятельности в рамках выбранно го направления подготовки и получение навыков применения теоретических знаний в практической деятельности. Вранее пройденном курсе «Введение в специальность» произошло общее знаком ство с возможными направлениями деятельности специалистов в области приклад ной математики и получен опыт оформления работ (реферата), который полезен при оформлении отчета по практике. В рамках освоенного курса «Введение в информационные технологии» изучены основные возможности языка программирования C++ и сформированыбазовые уме ния в области программирования на C++. Задачей практики является закрепление соответствующих знаний и умений и овладение навыками разработки программ на языке C++, реализующих заданные алгоритмы. Кроме того, практика предполагает формирование умений работы с системами компьютерной алгебры и уяснение раз личий в принципах построения алгоритмов решения задач при их реализации на языках программирования высокого уровня (к которым относится язык C++) и на языках функционального программирования (реализуемых системами компьютер ной алгебры)

2. История задачи

Задача коммивояжера, сформулированная Карлом Менгером в 1930 году на основе более ранних идей о гамильтоновых циклах, является эталонной NP-трудной проблемой оптимизации, где требуется найти кратчайший замкнутый маршрут через все города. Её осробеность в сочетании простоты формулировки с вычислительной сложностью: такие локальные методы, как восхождение к вершине, легко находят хорошие решения, нахождение действительно оптимального пути является достаточно сложной задачей. Учитывая эти свойства, начиная со второй половины XX века исследование задачи коммивояжёра имеет не столько практический смысл, сколько теоретический в качестве модели для разработки новых алгоритмов оптимизации. Многие современные распространенные методы дискретной оптимизации, такие как метод отсечений, ветвей и границ и различные варианты эвристических алгоритмов, были разработаны на примере задачи коммивояжёра.

3. Поиск восхождением к вершине

Поиск восхождением к вершине – это алгоритм локального поиска, являющемся итеративным методом, который начинается с произвольного решения задачи, нга каждой итерации ищется лучшее решение путём пошагового изменения одного из элементов решения. Если решение даёт улучшение, то алгоритм переходит на следующую итерацию, в которой начальным решением будет только что полученное. Этот процесс продолжается пока не будет достигнут момент, в котором улучшение найти не удаётся.

В контексте задачи коммивояжера решением является последовательность посещенных городов, а ближайшие решения, это решения полученные перестановкой двух городов в цепочке. Таким образом в задачи с n городами будет $\frac{n(n-1)}{2}$ ближайших решений. В стандартной реализации итерация заканчивается на первом ближайшем решении, показавшем улучшение, но в steepest ascent hill climbing сосед, показавший лучшее решение.

Главной проблемой даного алгоритма являются остановка в локальных экстремумах. Это происходит потому, что поиск восхождением к вершине никогда не допускает временного ухудшения решения, которое могло бы в перспективе привести к нахождению лучшего маршрута. Для преодоления этого ограничений была разработана модификация метода – многостартовый Hill Climbing.В эом подходе многократно запускается алгоритм из разных начальных точек, и выбирается лучший результат, что повышает шанс попасть в абсолютный минимум. Но даже модефицированная вариацияи не может гарантировать нахождне оптимального решения. Несмотря на свои недостатки, метод восхождения к вершине остаётся важным инструментом в оптимизации благодаря простоте реализации и высокой скорости работы, особенно в случаях, когда требуется быстро получить хорошее, но не обязательно идеальное решение.

4. Реализация поиска восхождением к вершине

4.1. Реализация на языке C++

В коде используется STL (<vector>, <algorithm>, <random>). Генерация случайных чисел выполняется через std::mt19937 с сидом от std::random_device, что обеспечивает высокую случайность. Главный цикл в hillClimbingTSP() перебирает соседей через std::swap.

5. Метод отсечений (алгоритм Гомори)

Алгоритм Гомори — алгоритм, который используется для решения полностью целочисленных задач линейного программирования. Суть алгоритма в том, что уже имея не цлочисленное решение задачи, он формирует новые условия, которые добавляются к основной задачие. Если если с этими условиями решение все еще не целочислено, то составляются еще условия, пока решение не будет целочисленным.

Условия задачи коммивояжера с n городами можно сформулровать как задачу линейного програмирования. D_{ij} – растояние от города i до j (млжет быть не симметричным), $x_[ij]$ – значения элемента в графе свзяности графа движение между городами. У пути, подходящего под ответ долже быть 1 вход в каждый город и 1 выход, это значит что сумма во всех строчках и всех столбцах должна равняться 1. Тогда целевая функция и условия будут выглядить так:

$$\sum_{i=1,j=1}^{n} D_{ij}x_{ij} \to \min,$$

$$\sum_{i=1}^{n} x_{ik} = 1 \quad , \quad k = \overline{1,n},$$

$$\sum_{j=1}^{n} x_{kj} = 1 \quad , \quad k = \overline{1,n}.$$

Данные условия являются условиями задачи линейого програмирования, которую мы будем решать симпликс методом. Симплекс-метод представляет собой итеративный процесс, основанный на последовательном улучшении допустимого решения. Алгоритм начинается с приведения задачи к каноническому виду, где все ограничения являются уравнениями, а переменные неотрицательны. Для этого вводятся дополнительные переменные, преобразующие неравенства в равенства. Исходная задача максимизации может быть легко переформулирована как задача минимизации, и наоборот, путем изменения знака целевой функции.

На следующем этапе строится начальная симплекс-таблица, содержащая коэффициенты системы ограничений и целевой функции. Критерием оптимальности в задаче минимизации служит неотрицательность всех коэффициентов в строке целевой функции. Если это условие не выполняется, выбирается ведущий столбец с наименьшим отрицательным значением, что соответствует направлению наибольшего улучшения целевой функции.

Затем определяется ведущая строка путем вычисления минимального отношения правых частей уравнений к положительным коэффициентам выбранного столбца.

Это отношение указывает на разрешающий элемент, вокруг которого выполняется поворот таблицы. В процессе итерации ведущая строка нормируется на разрешающий элемент, а остальные строки преобразуются таким образом, чтобы обнулить все элементы ведущего столбца, кроме разрешающего элемента. Это соответствует переходу к новой вершине многогранника допустимых решений с улучшенным значением целевой функции.

Процесс повторяется до тех пор, пока не будут выполнены условия оптимальности либо не обнаружится неограниченность целевой функции.

Симплекс-метод, несмотря на свою вычислительную сложность в худшем случае, остается одним из наиболее эффективных и широко применяемых алгоритмов линейного программирования благодаря высокой скорости работы на практике и возможности адаптации к различным типам задач.

Решение при таких условиях мжет дать ответ, имеющий подциклы, что противоречит условию задачи. Для устранения этого вводится доп условия на переменные, находящиеся в цикле:

$$\sum x_{ij} = len(S) \quad , \quad x_{ij} \in S.$$

Здесь S – подцикл, $len(S)$ – длина подцикла.

Решив задачу симплекс методом с дополненными условиями, может получится не целочисленное решение. Тут и нужен алгоритм Гамори. Пусть в оптимальном решении задачи ЛП некоторая базисная переменная x_{ij} принимает дробное значение. Тогда соответствующее уравнение симплекс-таблицы имеет вид:

$$x_i + \sum_{j \in N} a_{ij} x_j = b_i$$

где N – множество индексов небазисных переменных, b_i – дробное число. Отсечение Гомори для этого уравнения записывается как:

$$\sum_{j \in N} \left(a_{ij} - \lfloor a_{ij} \rfloor\right) x_j \geq b_i - \lfloor b_i \rfloor$$

Отсечение Гомори гарантирует, что текущее нецелочисленное решение перестанет быть допустимым, но все целочисленные точки останутся в области решений. Алгоритм продолжает добавлять отсечения до тех пор, пока не будет найдено целочисленное решение или не будет доказана его несуществование.

6. Реализация метода отсечений

6.1. Реализация на языке C++

Не были использованные готовые библеотеки, поэтому симплик метод, алгоритм гамори и алгоритм по нахождения подциклов и добавлению ддополнительных условий были реализованы с нуляю. Был реализрван класс SimplexMethod в котором хранились все компоненты симплекс таблицы, базисные переменные и количество исходных и дополнительных переменных.

Список использованных источников

1. Эльсгольц, Лев Эрнестович. Дифференциальные уравнения и вариационное исчисление. Наука, 1969. 424 с.
2. Nahin, Paul J. Chases and escapes: the mathematics of pursuit and evasion. Princeton University Press, 2012. 285 с.