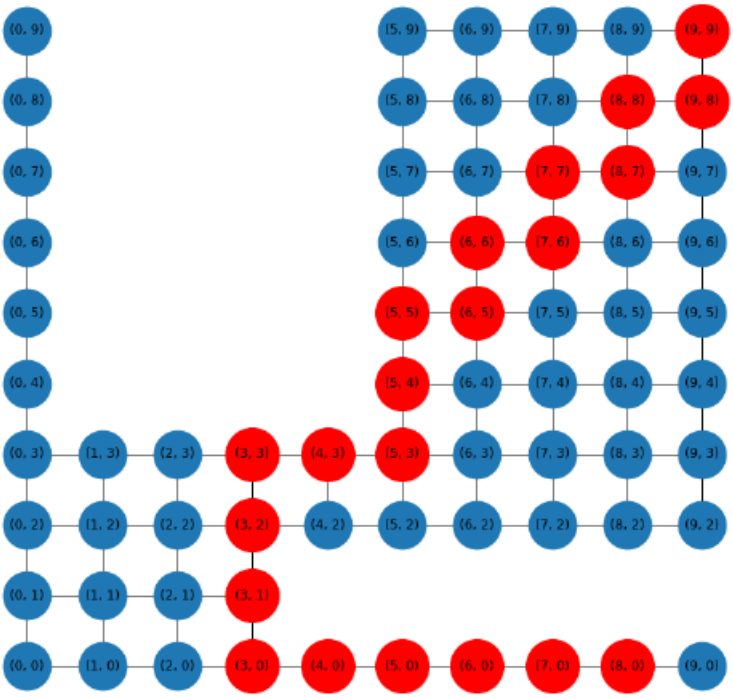
УНИВЕРСИТЕТ ИТМО

П.В. Чунаев, К.О. Боченина

АНАЛИЗ И РАЗРАБОТКА АЛГОРИТМОВ

Учебно-методическое пособие



Санкт-Петербург
2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
УНИВЕРСИТЕТ ИТМО

П.В Чунаев, К.О. Боченина

АНАЛИЗ И РАЗРАБОТКА АЛГОРИТМОВ

Учебно-методическое пособие

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ
В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки 09.04.02 Информационные системы и технологии в качестве учебно-методического пособия для реализации основных профессиональных образовательных программ высшего образования магистратуры

Санкт-Петербург
2020

Чунаев П.В., Боченина К.О. Анализ и разработка алгоритмов: учебно-методическое пособие. — СПб: Университет ИТМО, 2020. — 33 с.

Рецензент: Гладилин Петр Евгеньевич, кандидат физико-математических наук, научный сотрудник Национального центра когнитивных разработок Университета ИТМО.

Данное пособие предназначено для слушателей курса «Анализ и разработка алгоритмов» и содержит учебно-методические рекомендации к выполнению лабораторных по курсу.

В пособии представлены семь лабораторных работ, направленных на практическое усвоение теории анализа и разработки алгоритмов. Особое внимание уделено анализу и практической реализации специальных алгоритмов, предназначенных, прежде всего, для решения задач машинного обучения. В связи с этим в предлагаемых лабораторных работах рассматриваются алгоритмы оптимизации многомерных функций и их приложения к анализу данных, а также алгоритмы на графах и их приложения к анализу сетей различной природы.

Учебно-методическое пособие предназначено для магистрантов по направлению подготовки 09.04.02 Информационные системы и технологии.

УНИВЕРСИТЕТ ИТМО

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009-ом году статус национального исследовательского университета. С 2013-го года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научнообразовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

Содержание

Введение

В настоящем пособии представлены учебно-методические рекомендации к выполнению лабораторных работ по курсу «Анализ и разработка алгоритмов» в соответствии с курсом лекций, читаемым авторами магистрантам Университета ИТМО, обучающимся по направлению подготовки 09.04.02 Информационные системы и технологии, в том числе по программам «Большие данные и машинное обучение» и «Цифровое здравоохранение».

Предлагаемые лабораторные работы направлены на практическое усвоение магистрантами теории анализа и разработки алгоритмов. Особое внимание уделено анализу и практической реализации специальных алгоритмов, предназначенных, прежде всего, для решения задач машинного обучения. В частности, магистрантам предстоит рассмотреть алгоритмы оптимизации многомерных функций и их приложения к анализу данных, а также алгоритмы на графах и их приложения к анализу сетей различной природы.

В данном пособии для каждой лабораторной работы формулируется ее цель, а также ставятся задачи, которые требуется решить для достижения цели. Далее приводятся краткие теоретические сведения, необходимые для выполнения лабораторной работы, и даются примеры ожидаемых результатов. Указанные сведения всюду снабжены библиографическими ссылками на рекомендуемую литературу для более детального изучения рассматриваемых тем. После описания каждой лабораторной работы также приведены вопросы для самоконтроля.

Предполагается, что для выполнения лабораторных работ магистрант может использовать любой доступный ему язык программирования, позволяющий решать поставленные задачи. При этом важно учитывать возможность вывода соответствующих графических материлов. По опыту, для выполнения лабораторных работ магистранты зачастую используют Python и соответствующие открытые библиотеки.

По итогам выполнения каждой лабораторной работы магистрант должен подготовить стандартизированный отчет. Образец титульного листа и общая структура отчета представлены в Приложении А к данному пособию. Особо отметим необходимость использования в отчете содержательных и информативных графических материалов (схем, графиков, таблиц и др.) с соответствующими подписями и пояснениями.

Отчет о выполнении лабораторной работы сначала проходит проверку на соответствие формальным требованиям оформления и содержательности, а также проверку на наличие в нем признаков плагиата или машинного перевода. Если отчет, например, не соответствует указанным

требованиям или в нем обнаружен плагиат, то такой отчет отправляется на доработку. Автор отчета о выполнении лабораторной работы также должен защитить полученные им результаты в устной или письменной форме.

Авторы выражают благодарность В. Власенко, Т. Градову, В. Провалову и И. Сафиуллину за предоставленные иллюстрации к лабораторным работам.

Лабораторная работа №1
Эмпирический анализ временной сложности алгоритмов

Цель работы

Эмпирический анализ временной сложности алгоритмов.

Задание

Для каждого n от 1 до 2000 произведите для пяти запусков замер среднего машинного времени исполнения программ, реализующих нижеуказанные алгоритмы и функции. Изобразите на графике полученные данные, отражающие зависимость среднего времени исполнения от n. Проведите теоретический анализ временной сложности рассматриваемых алгоритмов и сравните эмпирическую и теоретическую временные сложности.

I. Сгенерируйте n-мерный случайный вектор $v = [v_1, v_2, \ldots, v_n]$ с неотрицательными элементами. Для полученного вектора v осуществите подсчет функций и реализацию алгоритмов:

1. $f(v) = 1$ (постоянная функция);
2. $f(v) = \sum_{k=1}^{n} v_k$ (сумма элементов);
3. $f(v) = \prod_{k=1}^{n} v_k$ (произведение элементов);
4. полагая, что элементы v – коэффициенты многочлена P степени $n-1$, вычислите значение $P(1,5)$ путем прямого (наивного) вычисления $P(x) = \sum_{k=1}^{n} v_k x^{k-1}$ для $x = 1,5$ (т.е. оценивая каждый член по одному) и методом Горнера представление полинома [10]:

$$P(x) = v_1 + x(v_2 + x(v_3 + \cdots)), \qquad x = 1,5;$$

5. алгоритм сортировки пузырьком (Bubble sort) элементов v;
6. алгоритм быстрой сортировки (Quick sort) элементов v;
7. гибридный алгоритм сортировки Timsort элементов v.

II. Сгенерируйте случайные матрицы A и B размером $n \times n$ с неотрицательными элементами. Найдите обычное матричное произведение матриц A и B.

Краткие теоретические сведения и примеры ожидаемых результатов

Временная сложность – это характеристика алгоритма, призванная давать представление о количестве времени, требуемом для работы алгоритма на определенном объеме данных [6, 10, 13]. Оценка временной сложности алгоритма осуществляется путем подсчета количества элементарных операций (сложений, умножений и т.д.), выполняемых алгоритмом для заданного объема данных. При этом предполагается, что выполнение каждой элементарной операции требует фиксированного количества времени.

В связи с тем, что время работы алгоритма может быть различно для данных разного объема, временную сложность принято определять с помощью функции T, отражающей зависимость времени работы алгоритма $T(n)$ от объема n входных данных. При этом особое внимание уделается асимптотическому поведению $T(n)$ при $n \to \infty$. При анализе асимптотического поведения $T(n)$ естественным образом учитываются только слагаемые самого высокого порядка, причем без внимания к их константным множителям. Поэтому чаще всего временную сложность $T(n)$ записывают в формате O-большое. В таких терминах утверждение о том, что временная сложность некоторого алгоритма равна $O(t(n))$, где $t(n) > 0$ – некоторая функция, означает, что с увеличением объема n входных данных время работы алгоритма будет асимптотически возрастать не быстрее, чем $C \cdot t(n)$ с некоторой фиксированной константой $C > 0$. Например, $O(n^2)$ – временная сложность алгоритма сортировки пузырьком элементов действительного вектора v размерности n, а $O(n^3)$ – временная сложность алгоритма обычного умножения двух матриц размера $n \times n$. В [12] приводятся временные сложности многих стандартных алгоритмов.

Оценка временной сложности алгоритмов не всегда проста; для некоторых алгоритмов временные сложности по-прежнему неизвестны. (Это замечание, конечно, не касается алгоритмов, рассматриваемых в данной лабораторной работе, – их временные сложности могут быть оценены [6, 10, 13].) Для того, чтобы все-таки получить представление о временной сложности алгоритма, можно применять эмпирический подход [6]. Он состоит в проведении серии замеров времени работы алгоритма при изменении объема входных данных. Например, для алгоритма, принимающего на вход векторы размерности n, можно замерять машинное время его работы при n, скажем, от 1 до 10^5 с шагом 10. При этом предполагается, что алгоритм запускается в одинаковых условиях, в частности, на одном и том же компьютере, не выполняющем каких-либо дополнительных

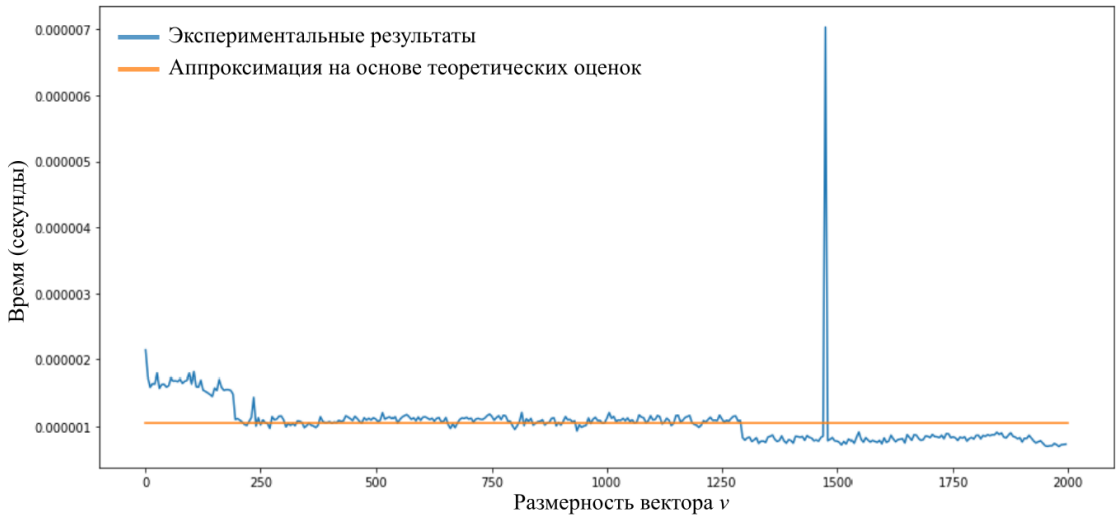


Рис. 1: Эмпирическая и теоретическая временные сложности алгоритма, реализующего функцию $f(v) = 1$.

вычислительных процессов, способных существенно повлиять на время работы рассматриваемого алгоритма. Наличие одинаковых условий при каждом замере принципиально важно для качества полученных результатов. Стоит отметить, что и при использовании одного и того же компьютера достичь этого не всегда удается – влияние на результат могут оказывать многочисленные факторы, в том числе фоновые процессы системы. Разумный компромисс в такой ситуации – усреднять замеры времени по нескольким запускам для одного и того же объема данных.

На рисунках 1 и 2 приведены примеры эмпирического исследования временной сложности алгоритма, реализующего функцию $f(v) = 1$, и алгоритма сортировки пузырьком элементов вектора v при разной размерности входного вектора v. Легко видеть, что эмпирически полученные кривые могут допускать даже значительные отклонения от теоретических кривых, но в среднем вполне очевидно соответствие между экспериментом и теорией.

Вопросы для самоконтроля

1. Обозначим количество вершин в графе через n. Что означает выражение $O(n)$, описывающее временную сложность некоторого алгоритма на графах?

2. С чем связана необходимость усреднять по нескольким запускам замеры времени работы алгоритма при эмпирическом анализе временной сложности алгоритма?

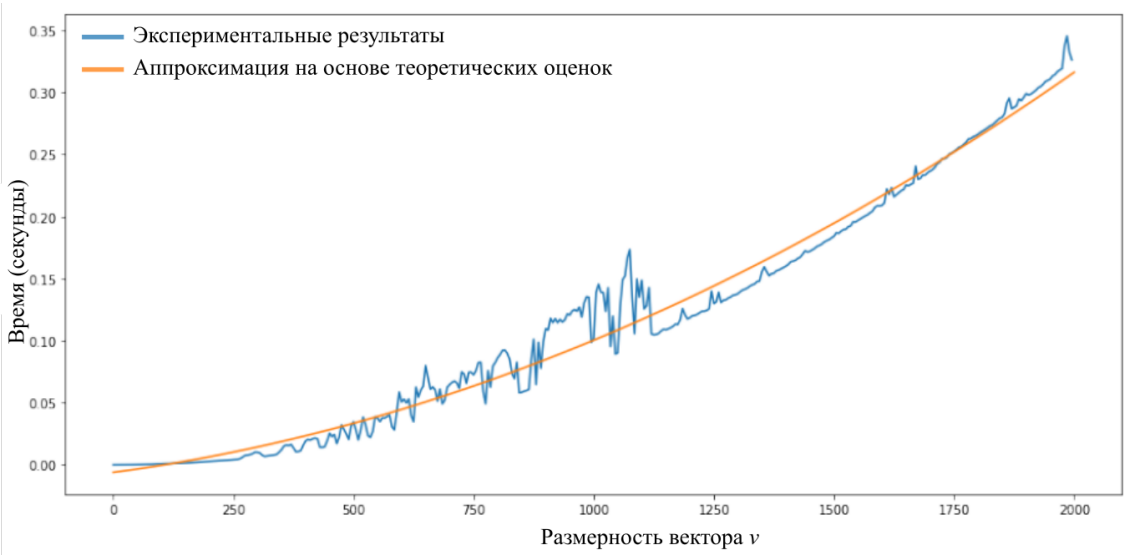


Рис. 2: Эмпирическая и теоретическая временные сложности алгоритма сортировки пузырьком элементов вектора v.

Лабораторная работа №2
Алгоритмы безусловной нелинейной оптимизации. Прямые методы

Цель работы

Применение прямых методов (одномерные методы перебора, дихотомии, золотого сечения; многомерные методы перебора, Гаусса, Нелдера-Мида) в задачах безусловной нелинейной оптимизации.

Задание

I. Реализуйте одномерные методы перебора, дихотомии и золотого сечения для приближенного (с точностью $\varepsilon = 0,001$) поиска x: $f(x) \to \min$ для следующих функций и областей допустимых значений:

1. $f(x) = x^3$, $x \in [0, 1]$;
2. $f(x) = |x - 0,2|$, $x \in [0, 1]$;
3. $f(x) = x \sin \frac{1}{x}$, $x \in [0.1, 1]$.

Подсчитайте количество вычислений функции f и количество произведенных итераций для каждого метода и проведите анализ полученных результатов. Объясните различия в полученных результатах, если таковые имеются.

II. Сгенерируйте случайные значения $\alpha \in (0, 1)$ и $\beta \in (0, 1)$. С использованием этих значений сгенерируйте массив зашумленных данных (x_k, y_k) для $k = 0, \ldots, 100$ по следующему правилу:

$$y_k = \alpha x_k + \beta + \delta_k, \quad x_k = \frac{k}{100},$$

где $\delta_k \sim N(0, 1)$ – значения случайной величины со стандартным нормальным распределением. Аппроксимируйте полученные данные линейной и рациональной функциями:

- $F(x, a, b) = ax + b$ (линейная аппроксимирующая функция);
- $F(x, a, b) = \frac{a}{1+bx}$ (рациональная аппроксимирующая функция),

с помощью метода наименьших квадратов [8] путем численной (с точностью $\varepsilon = 0,001$) минимизации функции

$$D(a,b) = \sum_{k=0}^{100} (F(x_k, a, b) - y_k)^2.$$

Для решения задачи минимизации используйте методы перебора, Гаусса и Нелдера-Мида. При необходимости самостоятельно задайте начальные приближения и прочие параметры методов. На графиках (отдельно для каждой *аппроксимирующей функции*) изобразите массив сгенерированных данных и графики аппроксимирующих функций, полученных с помощью указанных методов численной оптимизации. Проведите анализ полученных результатов.

Краткие теоретические сведения и примеры ожидаемых результатов

Методы оптимизации – это численные методы нахождения оптимальных (в некотором смысле) значений целевых функций, например, в рамках математических моделей тех или иных процессов [2, 18]. Методы оптимизации широко применяются при анализе данных и в машинном обучении [1, 2, 18].

Пусть задана целевая функция $f = f(x)$, где x – это, вообще говоря, многомерный вектор из некоторого подмножества Q евклидова пространства $\mathbb{R}^m$. Подможество Q может быть как ограниченным, так и, в частности, совпадать со всем пространством $\mathbb{R}^m$. Далее для краткости будем рассматривать только задачу минимизации функции f на множестве Q (от минимизации можно перейти к максимизации, рассматрев $F(x) = -f(x)$ вместо $f(x)$).

Под решением задачи оптимизации (минимизации) $f(x) \to \min_{x \in D}$ будем подразумевать нахождение $x^* \in Q$ такого, что $f(x^*) = \min_{x \in Q} f(x)$. Для x^* существует специальное обозначение: $x^* = \arg\min_{x \in Q} f(x)$. Если x^* найдено, то, естественно, можно найти и $f(x^*)$.

Указанная формулировка задачи оптимизации подразумевает поиск глобального минимума f на Q. Однако ниже мы будем рассматривать и поиск локального минимума f на Q, когда значение $f(x^*)$ минимально только в некоторой окрестности точки x^*. Заметим, что зачастую поиск локального минимума значительно проще, чем глобального.

Если задача оптимизации включает дополнительные условия на x^* (в виде системы S уравнений и неравенств), то ее называют условной.

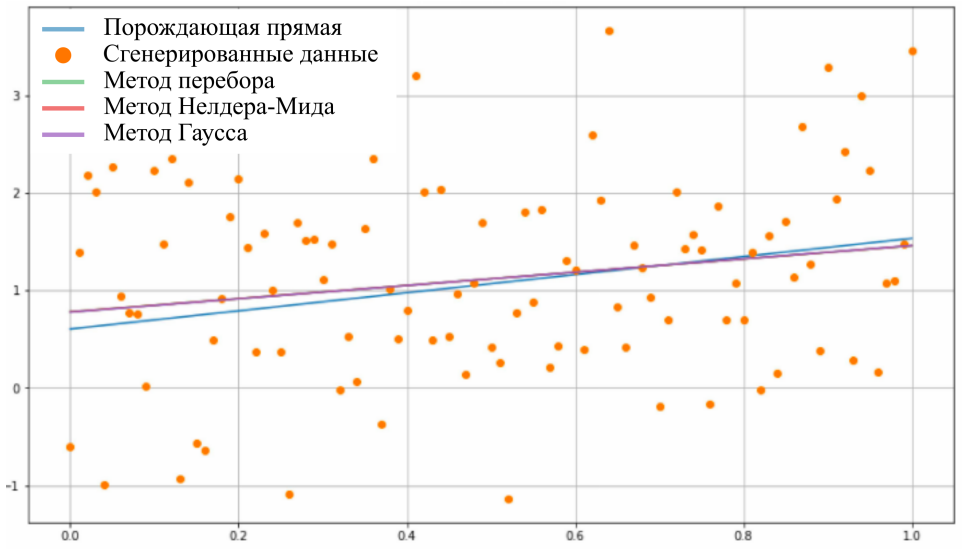


Рис. 3: Результаты применения методов перебора, Гаусса и Нелдера-Мида для решения указанной задачи линейной аппроксимации.

В противном случае задача оптимизации называется безусловной. Заметим еще, что в зависимости от вида f и S различают задачи линейной и нелинейной оптимизации. Здесь и ниже мы будем рассматривать только задачи *безусловной нелинейной оптимизации*. Это означает, что функция f в общем случае нелинейна, а условия S не накладываются.

В рамках этой лабораторной работы нам интересны *прямые методы оптимизации* (*методы оптимизации нулевого порядка*). По определению, они используют при поиске x^* только значения самой функции f, но не ее производных. Эти методы, в частности, применимы для непрерывных (и необязательно дифференцируемых) функций $f = f(x)$ одного переменного x на отрезке $Q = [0, 1]$. Вообще говоря, получается, что прямые методы можно использовать для довольно широкого класса функций f. Это, однако, компенсируется довольно низкой скоростью сходимости соответствующих итерационных процессов [1, 2, 7, 18].

Дадим несколько комментариев к задачам оптимизации в части I задания и методам перебора, дихотомии и золотого сечения. Подробное описание соответствующих алгоритмов можно найти в [2, 7, 18]. Обратите особое внимание на то, сколько вычислений функций f требуется на каждой итерации при использовании указанных алгоритмов. Заметьте также, что f в части I – непрерывные функции одного переменного x на указанных подмножествах $[0, 1]$, но для первых двух локальный минимум совпадает с глобальным, а для последней – имеют место многочисленные локальные минимумы. В последнем случае методы перебора, дихотомии и золотого сечения могут приводить к разным результатам.

Что касается части II задания, там требуется сгенерировать мас-

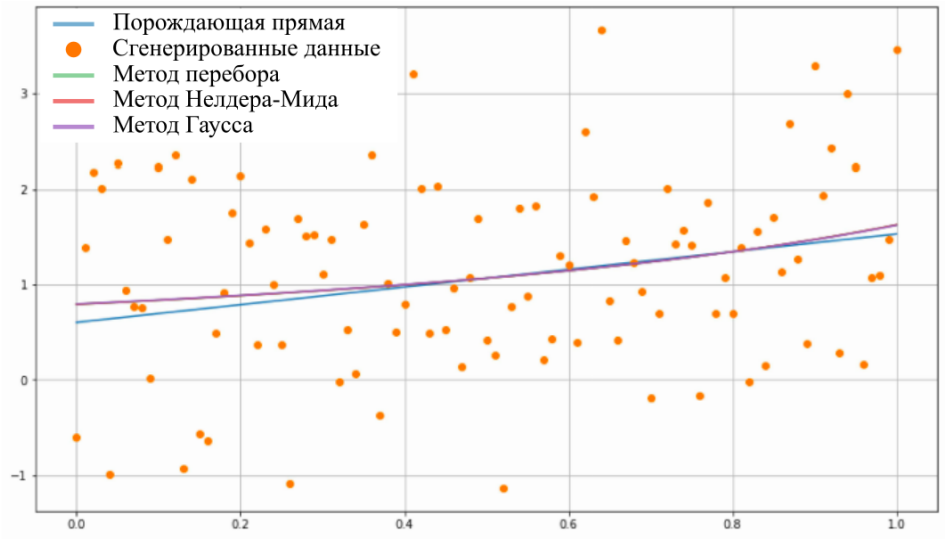


Рис. 4: Результаты применения методов перебора, Гаусса и Нелдера-Мида для решения указанной задачи рациональной аппроксимации.

сив *существенно* зашумленных данных и проверить, как для этих данных методы перебора, Гаусса и Нелдера-Мида (см. подробное описание в [1, 7, 17, 18]) решают задачи линейной и рациональной аппроксимации в смысле наименьших квадратов отклонений [8]. Хорошо известно, что указанная задача оптимизации, связанная с линейной аппроксимацией, имеет единственное решение, а потому стоит ожидать, что указанные методы будут давать весьма похожие оптимальные значения для a и b, вне зависимости от выбора начальных приближений (см. рисунок 3). В случае же рациональной аппроксимации возникают существенные нелинейности, а потому выбор начального приближения может существенно повлиять на результат. На рисунке 4 приведен пример, когда методы перебора, Гаусса и Нелдера-Мида все-таки приводят к довольно близким решениям задачи рациональной аппроксимации в смысле наименьших квадратов отклонений.

Вопросы для самоконтроля

1. Дайте определение прямых методов оптимизации.

2. Для выпуклой функции $f : [0, 1] \to \mathbb{R}$ какой из методов – перебора, дихотомии или золотого сечения – будет при фиксированном $\varepsilon > 0$ использовать больше значений f на отрезке $[0, 1]$? В чем принципиальное различие методов дихотомии и золотого сечения?

Лабораторная работа №3
Алгоритмы безусловной нелинейной оптимизации. Методы первого и второго порядка

Цель работы

Применение методов первого и второго порядка (градиентный спуск, метод сопряженных градиентов, метод Ньютона и алгоритм Левенберга-Марквардта) в задачах безусловной нелинейной оптимизации.

Задание

Сгенерируйте случайные значения $\alpha \in (0, 1)$ и $\beta \in (0, 1)$. С использованием этих значений сгенерируйте массив зашумленных данных (x_k, y_k) для $k = 0, \ldots, 100$ по следующему правилу:

$$y_k = \alpha x_k + \beta + \delta_k, \quad x_k = \frac{k}{100},$$

где $\delta_k \sim N(0, 1)$ – значения случайной величины со стандартным нормальным распределением. Аппроксимируйте полученные данные линейной и рациональной функциями:

- $F(x, a, b) = ax + b$ (линейная аппроксимирующая функция);
- $F(x, a, b) = \frac{a}{1+bx}$ (рациональная аппроксимирующая функция),

с помощью метода наименьших квадратов [8] путем численной (с точностью $\varepsilon = 0,001$) минимизации функции

$$D(a, b) = \sum_{k=0}^{100} (F(x_k, a, b) - y_k)^2.$$

Для решения задачи минимизации используйте градиентный спуск, метод сопряженных градиентов, метод Ньютона и алгоритм Левенберга-Марквардта. При необходимости самостоятельно задайте начальные приближения и прочие параметры методов. На графике (отдельно для каждой *аппроксимирующей функции*) изобразите массив сгенерированных данных и графики аппроксимирующих функций, полученных с помощью указанных алгоритмов численной оптимизации. Проведите анализ полученных результатов в терминах количества произведенных итераций. Сравните полученные результаты с результатами из лабораторной работы №2.

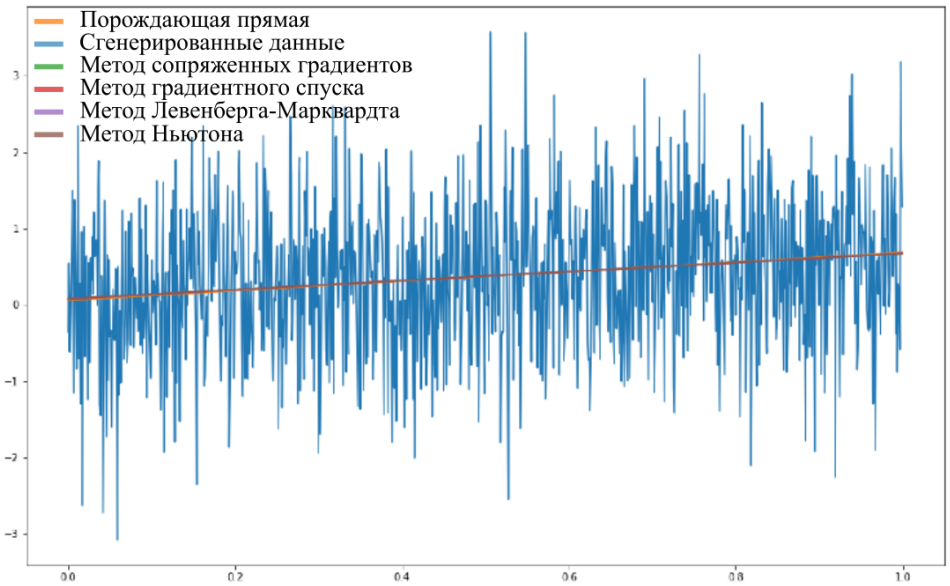


Рис. 5: Результаты применения градиентного спуска, метода сопряженных градиентов, метода Ньютона и алгоритма Левенберга-Марквардта для решения задачи линейной аппроксимации.

Краткие теоретические сведения и примеры ожидаемых результатов

Краткое введение в методы оптимизации и соответствующие определения и обозначения были даны в лабораторной работе №2. В текущей лабораторной работе рассматриваются методы оптимизации *первого и второго порядка*, т.е. методы, использующие для минимизации целевой функции f на множестве Q ее значения $f(x)$ и значения ее первой производной (градиента) и значения $f(x)$ и значения ее первой и второй производной (градиента, гессиана) соответственно. Естественно, подразумевается, что f непрерывно дифференцируема на Q достаточное количество раз.

В этой лабораторной работе используются методы первого порядка – градиентный спуск, метод сопряженных градиентов, и методы второго порядка – метод Ньютона и алгоритм Левенберга-Марквардта. Их подробное описание можно найти, например, в [1, 2, 7, 18].

Задача по сути та же, как в части II лабораторной работы №2. Именно, требуется сгенерировать массив *существенно* зашумленных данных и проверить, как для этих данных градиентный спуск, метод сопряженных градиентов, метод Ньютона и алгоритм Левенберга-Марквардта решают задачи линейной и рациональной аппроксимации в смысле наименьших квадратов отклонений [8]. Как уже упоминалось, задача оптимизации, связанная с линейной аппроксимацией, имеет единственное

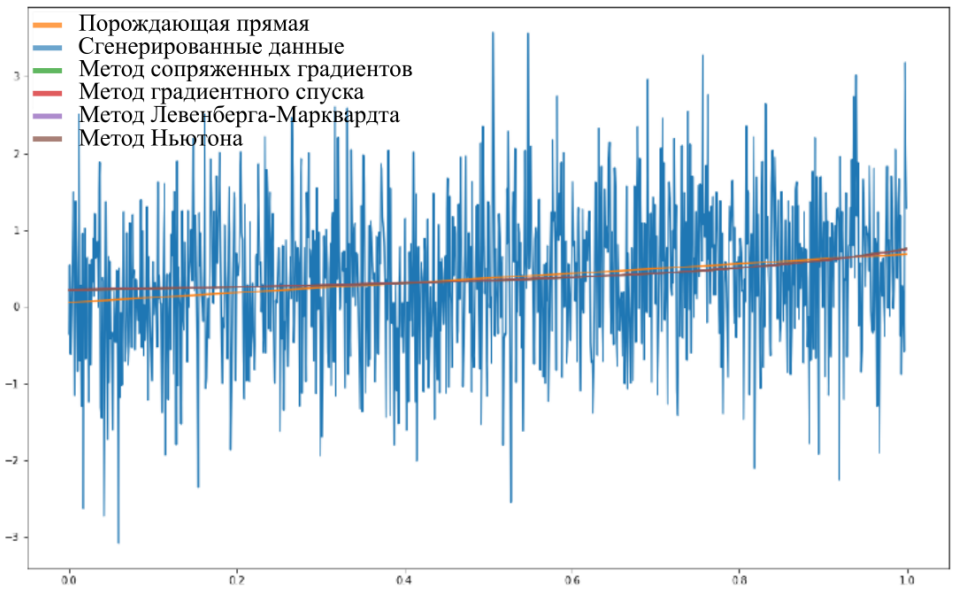


Рис. 6: Результаты применения градиентного спуска, метода сопряженных градиентов, метода Ньютона и алгоритма Левенберга-Марквардта для решения задачи рациональной аппроксимации.

решение, а потому стоит ожидать, что указанные методы будут давать весьма похожие оптимальные значения для a и b, вне зависимости от выбора начальных приближений (см. рисунок 5). В случае же рациональной аппроксимации возникают существенные нелинейности, и даже выбор начального приближение может существенно повлиять на результат. На рисунке 6 приведен пример, когда градиентный спуск, метод сопряженных градиентов, метод Ньютона и алгоритм Левенберга-Марквардта все-таки приводят к довольно близким решениям рассматриваемой задачи.

Вопросы для самоконтроля

1. Дайте определение методов оптимизации первого и второго порядка.

2. Пусть дана выпуклая функция $f : [0, 1] \to \mathbb{R}$ с непрерывными производными f' и f''. У какого из методов – покоординатного спуска или Ньютона – теоретическая скорость сходимости выше при оптимизации f на отрезке $[0, 1]$?

Лабораторная работа №4
Алгоритмы безусловной нелинейной оптимизации. Стохастические и метаэвристические алгоритмы

Цель работы

Использование стохастических и метаэвристических алгоритмов (имитация отжига, дифференциальная эволюция, метод роя частиц) в задачах безусловной нелинейной оптимизации и их экспериментальное сравнение с методами оптимизации нулевого и второго порядка (алгоритмами Нелдера-Мида и Левенберга-Марквардта).

Задание

Сгенерируйте зашумленные данные (x_k, y_k), где $k = 0, \ldots, 10^3$, следующим образом:

$$y_k = \begin{cases} -10^2 + \delta_k, & f(x_k) < -10^2 \\ f(x_k) + \delta_k, & |f(x_k)| \leq 10^2 \\ 10^2 + \delta_k, & f(x_k) > 10^2, \end{cases} \qquad x_k = \frac{3k}{10^3}, \quad f(x) = \frac{1}{x^2 - 3x + 2},$$

где δ_k $N(0, 1)$ – значения случайной величины со стандартным нормальным распределением. Аппроксимируйте полученные данные рациональной функцией

$$F(x, a, b, c, d) = \frac{ax + b}{x^2 + cx + d}.$$

с помощью метода наименьших квадратов [8] путем численной минимизации функции

$$D(a, b, c, d) = \sum_{k=0}^{1000} (F(x_k, a, b, c, d) - y_k)^2.$$

Для решения задачи минимизации используйте алгоритм Нелдера-Мида, алгоритм Левенберга-Марквардта и *хотя бы один из методов*: имитация отжига, дифференциальная эволюция или метод роя частиц. При необходимости задайте начальные приближения и другие параметры методов. Допускается не более 1000 итераций в каждом методе. Визуализируйте данные и аппроксимирующие кривые, полученные несколькими методами численной оптимизации, на одном графике. Рассчитайте сумму квадратов отклонений для каждого метода. Проанализируйте полученные результаты с точки зрения количества выполненных итераций и значений сумм квадратов отклонений.

Краткие теоретические сведения и примеры ожидаемых результатов

Краткое введение в методы оптимизации и соответствующие определения и обозначения были даны в лабораторной работе №2. Здесь мы фокусируемся на стохастических и метаэвристических алгоритмах оптимизации [11, 14, 15, 16].

Напомним, что стохастические (Монте-Карло) алгоритмы – это широкий класс алгоритмов, которые основаны на повторяющейся *случайной выборке* для решения задачи оптимизации. Эти методы наиболее полезны, когда невозможно или сложно применять другие (например, нет информации о дифференцируемости оптимизируемой функции или задача дискретная).

Метаэвристические алгоритмы – это *вдохновленные природными явлениями* алгоритмы, которые решают задачу оптимизации *методом проб и ошибок*. Метаэвристические методы, вообще говоря, не гарантируют, что будет найдено решение задачи оптимизации.

В рамках этой лабораторной работы рассматриваются методы имитации отжига, дифференциальной эволюции и роя частиц. Напомним, что имитация отжига – это метаэвристический алгоритм, который решает задачу оптимизации подобно процессу отжига в металлургии (нагрев и контролируемое охлаждение материала для увеличения размера его кристаллов и уменьшения дефектов). Дифференциальная эволюция – это метаэвристический алгоритм, который решает задачу оптимизации через эволюцию популяции *агентов*, то есть возможных решений, создавая новые поколения агентов путем объединения существующих и дальнейшего отбора лучших. Метод роя частиц – метаэвристический алгоритм, который решает задачу оптимизации путем итерационного изменения положения возможных решений (*частиц*) с определенной *скоростью*. На изменение положения каждой частицы влияют ее лучшее известное положение и лучшие известные положения других частиц.

Подробное описание указанных методов, а также введение в теорию стохастических и метаэвристических алгоритмов оптимизации можно найти в [11, 15, 14, 16].

Дадим несколько комментариев к заданию в рамках лабораторной работы №4. Там предлагается сначала сгенерировать зашумленные данные, порождаемые по существу рациональной функцией с разрывами на отрезке $[0, 1]$, и затем найти аппроксимирующую их рациональную функцию F соответствующего вида. Возникает крайне нелинейная задача численной оптимизации, где существенное влияние на результат может оказать выбранное начальное приближение. Для решения указан-

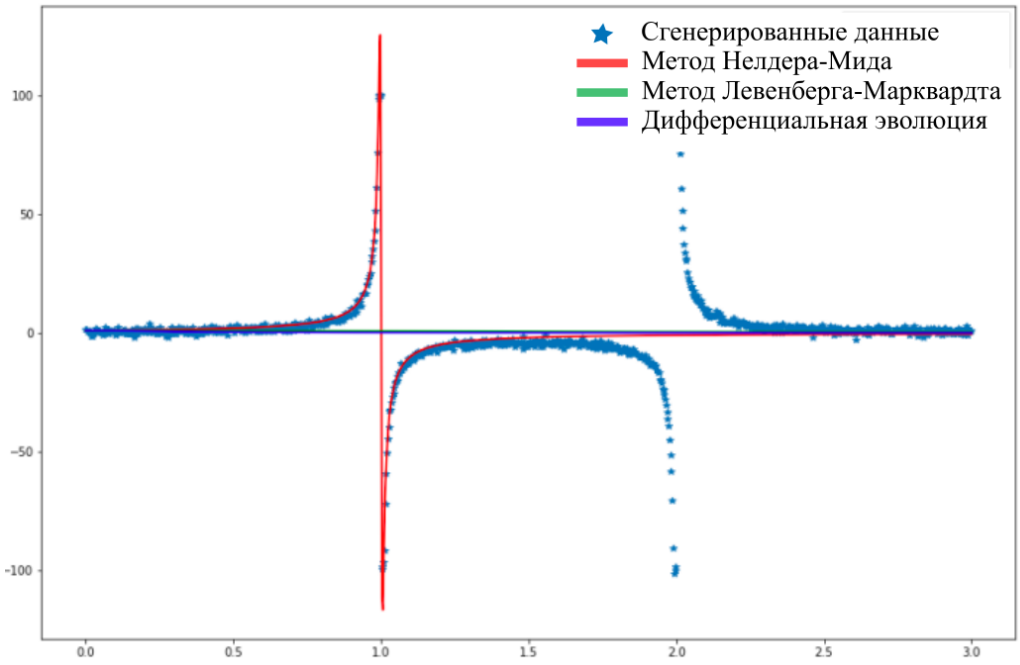


Рис. 7: Результаты применения методов Нелдера-Мида, Левенберга-Марквардта и дифференциальной эволюции для решения указанной задачи рациональной аппроксимации.

ной задачи предлагается использовать хотя бы один из вышеуказанных метаэвристических алгоритмов, а также метод Нелдера-Мида (прямой метод) и метод Левенберга-Марквардта (метод второго порядка). Необходимо также проанализировать полученные результаты. Один из примеров ожидаемых результатов дан на рисунке 7. Именно, там приведены аппроксимирующие F, найденные с помощью методов Нелдера-Мида, Левенберга-Марквардта и дифференциальной эволюции. Интересно, что среди этих методов только Нелдер-Мид выявил один из "разрывов" приближаемой функции. При этом остальные методы аппроксимировали данные почти прямой, что вызвало существенный рост суммы квадратов отклонений. Это, среди прочего, говорит о том, что указанные методы не всегда приводят к близким результатам и могут не находить искомое решение задачи оптимизации.

Вопросы для самоконтроля

1. Какие стохастические и метаэвристические алгоритмы нелинейной оптимизации вам известны? Назовите не менее трех.

2. Всегда ли такие алгоритмы находят глобальный экстремум целевой функции в задачах нелинейной оптимизации?

Лабораторная работа №5
Алгоритмы на графах. Введение в графы и основные алгоритмы на графах

Цель работы

Использование различных представлений графов и основных алгоритмов на графах (поиск в глубину и поиск в ширину).

Задание

I. Сгенерируйте случайную матрицу смежности для простого неориентированного невзвешенного графа со 100 вершинами и 200 ребрами (обратите внимание, что матрица должна быть симметричной и содержать только 0 и 1 в качестве элементов). Преобразуйте матрицу в список смежности. Визуализируйте граф и выведите несколько строк матрицы смежности и списка смежности. Для каких целей удобнее использовать каждое из представлений?

II. Используйте поиск в глубину и поиск в ширину, чтобы найти связанные компоненты графа и кратчайший путь между двумя случайными вершинами. Проанализируйте полученные результаты.

Краткие теоретические сведения и примеры ожидаемых результатов

Теория графов активно применяется при моделировании различных сложных систем, в том числе социальных, транспортных, коммуникационных и др. сетей. Инструменты теории графов позволяет проводить всесторонний аналих данных, представимых в виде графов. В рамках текущей лабораторной работы предлагается изучить на практике различные представления графа, а также стандартные алгоритмы обхода графа. Подробную информацию по этим вопросам, а также детальное введение в теорию графов и применяемые для анализа графов алгоритмы можно найти, например, в [4, 5, 9, 10, 13].

Напомним некоторые определения, которые могут понадобиться для решения поставленных задач. *Неориентированным графом* называют пару $G = (V, E)$, где $V = \{v_i\}$ – множество *вершин* (или *узлов*), и $E = \{e_{ij}\} = \{(v_i, v_j)\}$ – множество пар вершин, называемых *ребрами* (или *связями*). Число вершин обозначают через $|V|$, а число ребер – через $|E|$.

Ориентированный граф – это граф, в котором ребра имеют направления (ориентации). *Взвешенный граф* – это граф, в котором каждому ребру приписаны *веса*. *Простой граф* – это граф, в котором возможно только одно ребро между парой вершин. *Мультиграф* – это обобщение простого графа, при котором в графе возможны несколько ребер между парой вершин. *Полный граф* – это граф, в котором все вершины соединены ребром. *Путь* (*цепь*) в графе – это последовательность попарно различных ребер, соединяющих две различные вершины. *Длиной пути* (*цепи*) называют количество ребер (или сумму весов ребер) в пути (цепи). Вершины v_1 и v_2 в графе называются *связанными*, если существует путь из v_1 в v_2. В противном случае эти вершины называются *несвязанными*. *Связный граф* – это граф, в котором любая пара вершин связана. В противном случае граф называют *несвязным*. *Компонента связности графа* – это максимальный связный подграф графа.

Мы будем рассматривать простые неориентированные (взвешенные или невзвешенные) графы.

Перейдем к различным типам представления графа [10]. Первый их них – это *матрица смежности*, т.е. матрица, строки и столбцы которой индексируются вершинами и ячейки которой содержат булево значение (0 или 1), указывающее, являются ли соответствующие вершины смежными (для взвешенных графов вместо 1 стоят соответствующие веса). Матрица смежности (как $2D$ массив) требует $O(|V|^2)$ памяти. Пример матрицы смежности приведен на рисунке 8.

```
[[0. 1. 0. ... 1. 1. 1.]
 [1. 0. 0. ... 1. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [1. 1. 0. ... 0. 1. 0.]
 [1. 0. 0. ... 1. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]]
```

Рис. 8: Пример матрицы смежности.

Еще одним типом представления является *список смежности*, т.е. коллекции списков вершин, где для каждой вершины графа приведен список смежных ей вершин. Список смежности (как $1D$ массив списков) требует $O(|V| + |E|)$ памяти. Пример списка смежности приведен на рисунке 9.

Для *разреженного графа*, т.е. графа, в котором большинство пар вершин не связаны ребрами, $|E| \ll |V|^2$, список смежности значительно эффективнее для хранения, чем матрица смежности.

Для целей визуализации применяются и изображения графов в виде,

```
0 {1, 5, 8, 11, 16, 19, 20, 21, 22, 24, 25, 26, 28, 30, 34, 41, 42, 48, 49,
52, 53, 54, 56, 61, 63, 66, 67, 75, 77, 79, 80, 82, 86, 87, 90, 93, 94, 97,
98, 99}
1 {0, 5, 7, 11, 12, 16, 18, 20, 21, 23, 26, 29, 33, 37, 38, 39, 40, 47, 49,
50, 52, 57, 63, 65, 66, 71, 72, 74, 83, 84, 86, 87, 88, 91, 92, 93, 95, 97}
2 {4, 5, 7, 10, 12, 13, 14, 15, 16, 17, 20, 22, 23, 25, 29, 32, 33, 34, 35,
37, 38, 39, 41, 42, 43, 48, 49, 53, 55, 56, 58, 60, 61, 64, 67, 69, 70, 73,
74, 75, 77, 79, 81, 83, 84, 85, 86, 87, 88, 89}
3 {4, 8, 10, 11, 12, 14, 18, 19, 22, 24, 25, 26, 28, 29, 31, 35, 36, 43, 44,
45, 48, 49, 54, 55, 58, 61, 65, 66, 67, 68, 73, 74, 75, 77, 79, 84, 91, 92,
93, 95, 96}
4 {2, 3, 6, 8, 11, 12, 14, 16, 17, 20, 22, 23, 25, 32, 33, 35, 37, 38, 40, 4
3, 45, 53, 55, 59, 60, 61, 65, 66, 68, 69, 70, 77, 78, 79, 80, 83, 84, 85, 8
6, 87, 89, 92, 93, 94, 95, 99}
5 {0, 1, 2, 9, 11, 12, 17, 20, 22, 23, 25, 27, 28, 34, 36, 37, 39, 42, 44, 4
6, 47, 49, 54, 55, 56, 58, 59, 60, 63, 64, 66, 70, 77, 78, 80, 82, 83, 85, 8
8, 90, 91, 92, 93, 94, 98}
```

Рис. 9: Пример списка смежности.

как на рисунке 10. В общем случае задача информативной визуализации графов очень сложна.

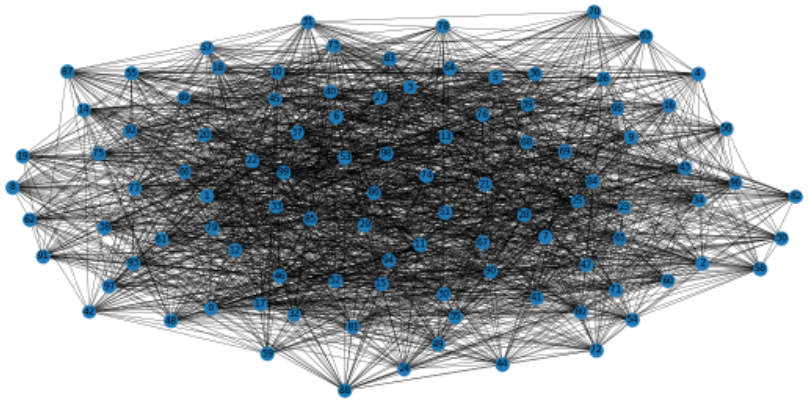


Рис. 10: Пример визуализации графа.

Перейдем к классическим алгоритмам обхода графа, рассматриваемым в рамках лабораторной работы №5. Их детальное описание можно найти в [13, 10].

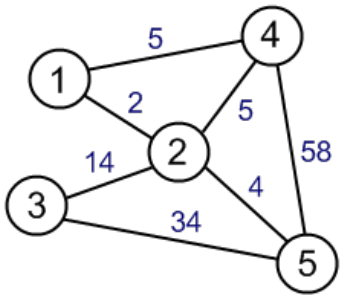
Поиск в глубину (Depth-first search, DFS) – это алгоритм обхода графа, который начинает обход с выбранной корневой вершины и идет «вглубь» графа, насколько это возможно, перед обходом из новой вершины. Его временная сложность равна $O(|V| + |E|)$. *Поиск в ширину* (Breadth-first search, BFS) – это алгоритм обхода графа, в котором обход начинается с выбранной корневой вершины и исследуются все смежные вершины на текущей «глубине», прежде чем перейти к вершинам на следующей «глубине». Этот алгоритм обхода использует стратегию, в некотором смысле противоположную DFS. Его временная сложность также

равна $O(|V| + |E|)$.

Оба алгоритма могут применяться для поиск кратчайшего пути между вершинами и поиска компонент связности графа.

Вопросы для самоконтроля

1. Охарактеризуйте граф, приведенный ниже: ориентированный/неориентированный; взвешенный/невзвешенный; связный/несвязный.



2. Одинакова ли временная сложность алгоритмов обхода графа в ширину и в глубину (при одинаковых способах хранения графа)?

Лабораторная работа №6
Алгоритмы на графах. Алгоритмы поиска пути на взвешенных графах

Цель работы

Использование алгоритмов поиска пути на взвешенных графах (алгоритм Дейкстры, A* и алгоритм Беллмана-Форда).

Задание

I. Сгенерируйте случайную матрицу смежности для простого неориентированного взвешенного графа из 100 вершин и 500 ребер с назначенными случайными положительными целыми весами (обратите внимание, что матрица должна быть симметричной и содержать только 0 и веса в качестве элементов). Используйте алгоритмы Дейкстры и Беллмана-Форда, чтобы найти кратчайшие пути между случайной начальной вершиной и другими вершинами. Измерьте время, необходимое для поиска путей каждому алгоритму. Повторите эксперимент 10 раз для одной и той же начальной вершины и рассчитайте среднее время, необходимое для поиска путей каждому алгоритму. Проанализируйте результаты.

II. Сгенерируйте сетку 10×10 с 30 ячейками-препятствиями. Выберите две случайные разрешенные ячейки и найдите кратчайший путь между ними, используя алгоритм A*. Повторите эксперимент 5 раз с разными парами ячеек. Проанализируйте результаты.

Краткие теоретические сведения и примеры ожидаемых результатов

Краткое введение в теорию графов и соответствующие определения и обозначения были даны в лабораторной работе №5. Детальное введение в теорию графов, а также описания рассматриваемых в рамках этой лабораторной работы алгоритмов на графам можно найти в [4, 5, 9, 10, 13].

Напомним, что *взвешенный граф* – это граф, в котором каждому ребру приписан *вес* (некоторое число). Для взвешенных графов требуются специальные алгоритмы обхода. Рассмотрим некоторые из них.

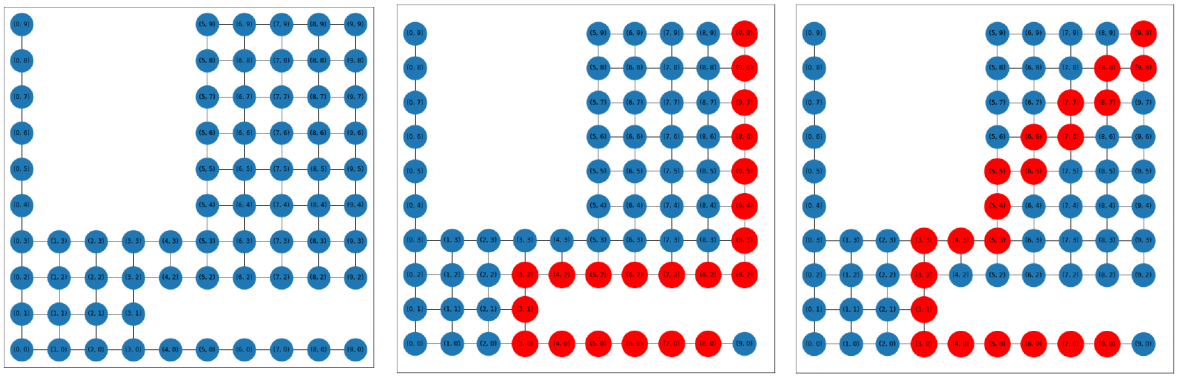
Алгоритм Дейкстры (DA) решает следующую задачу: для заданных графа (с *положительными* весами) и вершины-источника s, найти крат-

чайшие пути от s до остальных вершин. Основная идея алгоритма Дейкстры в том, что он генерирует дерево кратчайших путей (SPT) с корнем s путем обработки двух множеств: одно множество содержит вершины, включенные в SPT, другое – вершины, еще не включенные в SPT. На каждом шаге алгоритм находит вершину, не включенную в SPT и имеющую минимальное расстояние от источника. Сложность алгоритма оценивается от $O(|V| \log |V|)$ до $O(|V|^2)$ в зависимости от применяемой модификации.

Алгоритм A* решает следующую задачу: для данных графа (с *положительными* весами), источника s и цели t найти кратчайший путь от s до t. Основная идея алгоритма в том, что на каждой итерации он определяет, как продлить путь, исходя из стоимости текущего пути от s до точки продления и *оценки* стоимости пути от точки продления до t (это эвристика в алгоритме A*). Временная сложность алгоритма $O(|E|)$. Заметим, что из алгоритма A* получается DA, если вышеупомянутая оценка (эвристика) опущена.

В рамках лабораторной работы предлагается реализовать и применить алгоритм A* в контексте сетки с препятствиями, представимой в виде взвешенного графа. При этом эвристическая оценка стоимости пути на каждой итерации алгоритма рассчитывается без учета препятствий. На рисунке 11 (б, в) приведены примеры работы алгоритма A* для сетки (а) при различных эвристиках ($s = (8, 0)$, $t = (9, 9)$).

Алгоритм Беллмана-Форда (BFA) решает следующую задачу: для данного взвешенного графа (возможно, с *отрицательными* весами) и источником s найти кратчайшие пути от s до всех остальных вершин. Если граф содержит отрицательный цикл C_- (т.е. цикл, сумма ребер в котором отрицательна), достижимый из s, то кратчайшего пути не существует: *любой путь, имеющий вершину в* C_-*, может быть сделан*



(а) (б) (в)

Рис. 11: Примеры работы (б, в) алгоритма A* для сетки с препятствиями (а) при различных эвристиках ($s = (8, 0)$, $t = (9, 9)$).

короче при еще одном обходе C_-. В таком случае BFA сообщает об отрицательном цикле C_-.

Основная идея BFA состоит в следующем. На i-ой итерации BFA вычисляет кратчайшие пути, которые имеют не более i ребер. Поскольку в любом простом пути не более $|V|-1$ ребер, $i = 1, \ldots, |V|-1$. Предполагая, что нет C_-, если мы вычислили кратчайшие пути из не более чем i ребер, то итерация по всем ребрам гарантирует получение кратчайших путей из не более чем $i+1$ ребро. Чтобы проверить, есть ли отрицательный цикл C_-, BFA выполняет $|V|$-ую итерацию. Если хотя бы один из кратчайших путей становится короче, то имеет место отрицательный цикл C_-.

Временная сложность BFA весьма высока и оценивается как $O(|V||E|)$.

Вопросы для самоконтроля

1. Какой алгоритм поиска наикратчайшего пути – Дейкстры или Беллмана-Форда – обычно применяют для работы со взвешенными графами с отрицательными весами и возможными отрицательными циклами?

2. Применим ли алгоритм A^* к взвешенному графу, ассоциированному с географической картой (вершины – города, веса на ребрах – расстояния между городами)? Для чего нужна эвристика в алгоритме A^*?

Лабораторная работа №7
Алгоритмы на графах. Инструменты для анализа сетей

Цель работы

Использование ПО Gephi для анализа сетей.

Задание

1. Загрузите и установите Gephi с https://gephi.org/.
2. Выберите сеть в базе данных https://snap.stanford.edu/data/ с числом узлов не более 10000. Вы можете выбрать тип и тематику сети по собственному усмотрению (не/взвешенная, не/ориентированная).
3. При необходимости измените формат данных на тот, с которым работает Gephi (.csv, .xls, .edges и т.д.).
4. Импортируйте и обработайте данные сети в Gephi. Проверьте корректность импорта данных.
5. Получите раскладку двух разных типов для графа сети.
6. Рассчитайте доступные показатели сети в разделе Статистика Gephi.
7. Проанализируйте результаты для выбранной сети.

Краткие теоретические сведения и примеры ожидаемых результатов

Для выполнения этой лабораторной работы понадобятся дополнительные опеределения из теории графов. Напомним, что краткое введение в теорию графов и другие определения и обозначения были даны в лабораторной работе №5. Больше теоретического материала по теме можно найти в [4, 5, 9, 13].

Введем характеристики степеней вершин невзвешенного графа:

- $d(v)$, *степень* v, – количество входящих и исходящих ребер вершины v;

- $d_{\mathsf{in}}(v)$, *полустепень захода* v, – количество входящих ребер вершины v;
- $d_{\mathsf{out}}(v)$, *полустепень исхода* v, – количество исходящих ребер вершины v;
- $\overline{d} = \frac{1}{|V|}\sum_{v\in V} d(v)$ – *средняя степень вершин.*

Для взвешенного графа могут быть введены аналогичные величины, с той лишь разницей, что в них учитывается не количество (входящих/исходящих) ребер, а сумма весов на (входящих/исходящих) ребрах.

Теперь введем характеристики графа в смысле расстояний:

- $\mathsf{dist}(v, u)$ – расстояние (длина кратчайшего пути) между v и u (G – связный);
- *эксцентриситет* $\epsilon(v) = \max_{u\in V} \mathsf{dist}(v, u)$ – наибольшее расстояние между v и другими вершинами;
- *радиус* $r = \min_{v\in V} \epsilon(v)$ – наименьший эксцентриситет по всем вершинам;
- *диаметр* $D = \max_{v\in V} \epsilon(v)$ – наибольший эксцентриситет по всем вершинам, т.е. наибольшее расстояние между парой вершин;
- *средняя длина пути* $\ell = \frac{1}{|V|\cdot(|V|-1)}\sum_{v\neq u} \mathsf{dist}(v, u)$.

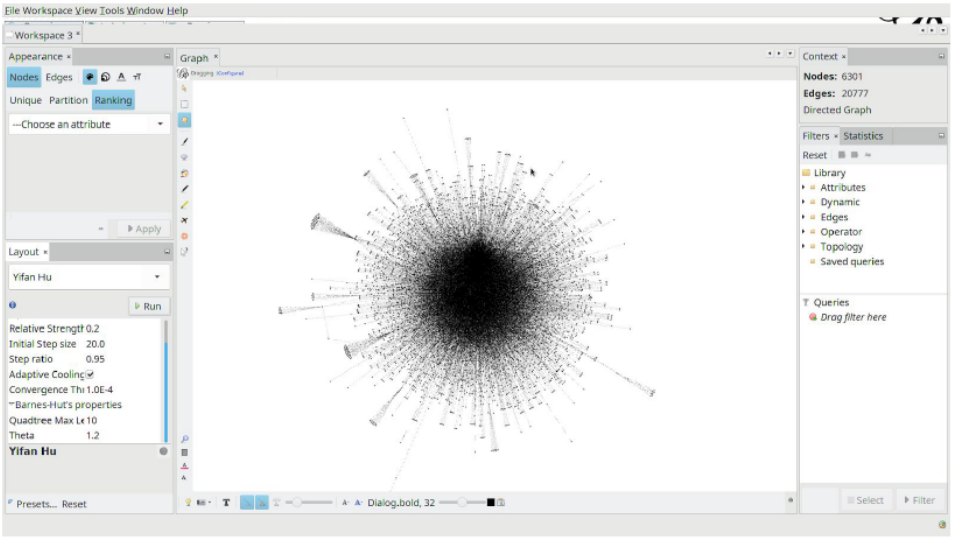
Далее, напомним, что *плотность* ρ графа – это частное $|E|$ и числа возможных ребер с тем же $|V|$, т.е. числа ребер в полном графе с $|V|$ вершинами:

$$\rho = \frac{2|E|}{|V|(|V|-1)}.$$

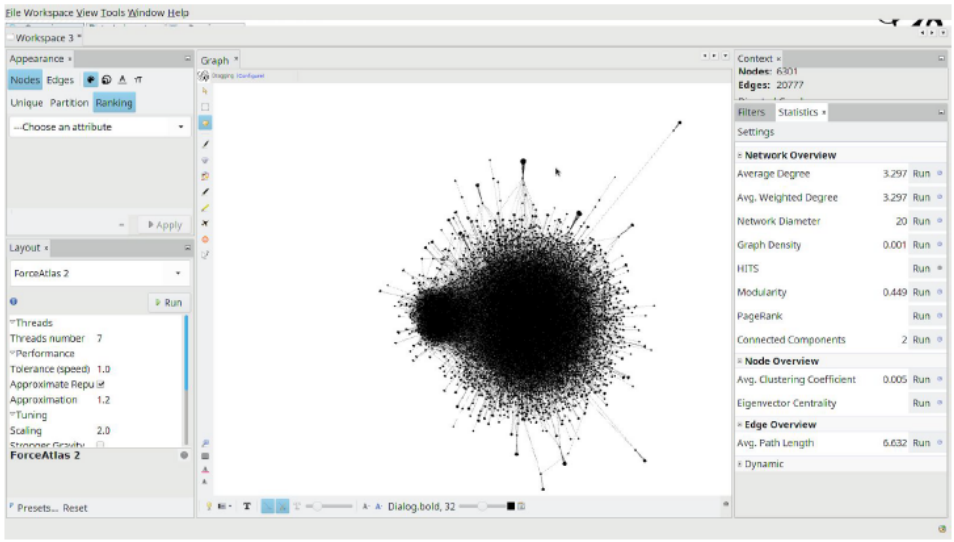
При $\rho \approx 0$ граф обычно называют *разреженным*. Заметим, что по определению плотность любого полного графа равна единице.

Приведенные выше и многие другие величины позволяют характеризовать граф с различных точек зрения. Это чрезвычайно полезно при анализе графов, моделирующих сложные сети реального мира.

Одним из удобных инструментов для проведения такого анализа, а также для визуализации графов является ПО Gephi, которое можно легко освоить по документации [3]. На рисунках 12 (а, б) приведен пример работы с Gephi для анализа и визуализации некоторого графа (моделирующего сеть реального мира), в частности, получены две его различные раскладки (визуализации). На рисунке 12 (б) справа также показаны значения для различных величин-характеристик графа, которые Gephi подсчитывает автоматически.



(а)



(б)

Рис. 12: Пример работы Gephi для некоторой сети реального мира.

Вопросы для самоконтроля

1. Какова плотность любого полного графа?

2. Дайте определение степени вершины, радиусу, диаметру, среднему пути и эксцентриситету графа.

Список литературы

[1] Бахвалов Н.С., Жидков Н.П., Кобельков Г.Г. Численные методы. М.: Лаборатория Базовых Знаний, 2000.

[2] Гилл Ф., Мюррей У., Райт М. Практическая оптимизация. М.: Мир, 1985.

[3] Документация Gephi https://gephi.org/users/.

[4] Емеличев В.А., Мельников О.И. Лекции по теории графов. М., 1990.

[5] Костюкова Н.И. Графы и их применение. Комбинаторные алгоритмы для программистов. М., 2007.

[6] Левитин, А.В. Алгоритмы: введение в разработку и анализ. М.: Издательский дом Вильямс, 2006.

[7] Лемешко Б.Ю. Методы оптимизации: Конспект лекций. Новосибирск: Изд-во НГТУ, 2009.

[8] Линник Ю.В. Метод наименьших квадратов и основы математико-статистической теории обработки наблюдений. М.: ФМЛ, 1958.

[9] Майника Э. Алгоритмы оптимизации на сетях и графах. М.: Мир, 1981.

[10] Макконелл Дж. Анализ алгоритмов. Вводный курс. М.: Техносфера, 2002.

[11] Пантелеев А.В., Скавинская Д.В. Метаэвристические алгоритмы глобальной оптимизации. М.: Вузовская книга, 2019.

[12] Big-O complexities of common algorithms used in Computer Science https://blogs.longwin.com.tw/wordpress/201608-bigoposter.pdf

[13] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. Introduction to Algorithms. Third Edition. The MIT Press, 2009.

[14] Glover, F., Kochenberger, G.A. Handbook of metaheuristics. 57. Springer, International Series in Operations Research and Management Science, 2003.

[15] Goldberg, D.E. Genetic Algorithms in Search, Optimization and Machine Learning. Kluwer Academic Publishers, 1989.

[16] Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P. Optimization by Simulated Annealing// Science. 220 (4598), pp. 671–680, 1983.

[17] Nelder, J.A., Mead, R. A simplex method for function minimization// Computer Journal. 7 (4), pp. 308–313, 1965.

[18] Nocedal, J., Wright, S.J. Numerical Optimization. Springer, 2006.

Приложение А

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИТМО»
(УНИВЕРСИТЕТ ИТМО)

Отчет
о выполнении лабораторной работы № X
«Тема работы»

Работу выполнил:
ст. группы <Номер группы>
Фамилия И.О.
Работу принял:
Фамилия И.О.

Санкт-Петербург
2020

Цель работы

Указывается цель лабораторной работы

Постановка задачи

Формулируются задачи, на решение которых направлена лабораторная работа

Краткая теоретическая часть

Приводятся краткие теоретические сведения, касающиеся содержания лабораторной работы, например, определения, описания алгоритмов, методические подходы к решению поставленных задач (не более 2 стр.).

Результаты

Приводятся результаты решения поставленных задач, в том числе графики и таблицы, а также кратко обсуждаются полученные результаты (не более 4 стр.)

Заключение

Делаются выводы о полученном решении поставленных задач и достижении цели лабораторной работы, дается оценка возможности применения приобретенных навыков и умений на практике

Приложение

Приводятся листинги написанных для выполнения лабораторной работы программ с комментариями

Чунаев Петр Владимирович
Боченина Клавдия Олеговна

АНАЛИЗ И РАЗРАБОТКА АЛГОРИТМОВ

Учебно-методическое пособие

В авторской редакции
Редакционно-издательский отдел
Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49