ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

Отчет о программном проекте

на тему: ПО для топологического анализа данных

(промежуточный, этап 1)

Выполнил:
Студент группы БПМИ225 — Подпись — С.А.Корняков (И.О.Фамилия)

29.01.2025
Дата

Принял:
Руководитель проекта — Качан Олег Николаевич
Имя, Отчество, Фамилия

-
Должность, ученое звание

Лаборатория ИИ Сбербанка
Место работы (Компания или подразделение НИУ ВШЭ)

Дата проверки _____ 2025 — Оценка (по 10-ти бальной шкале) — Подпись

Москва 2025

Содержание

1 Введение **3**

2 Требования к проекту **4**
2.1 Функциональные требования . . . 4
2.1.1 Генерация комплекса Вьеториса-Рипса по облаку точек . . . 4
2.1.2 Симплекс . . . 4
2.1.3 Симплициальный комплекс . . . 5
2.1.4 Фильтрация симплициального комплекса . . . 5
2.1.5 Столбцы и матрицы индексов . . . 5
2.1.6 Столбцы для быстрого сложения . . . 5
2.1.7 Диаграмма устойчивости . . . 6
2.1.8 Матрица Лапласиана . . . 6
2.1.9 Алгоритмы редуцирования . . . 6
2.1.10 Алгоритм поиска persistent HH . . . 6
2.2 Нефункциональные требования . . . 6
2.2.1 Описание программы и используемых библиотек . . . 6
2.2.2 Используемые библиотеки . . . 6
2.2.3 Политика обработки ошибок . . . 7

3 Теория и описание алгоритмов **7**
3.1 Дизайн решения . . . 7
3.2 Взаимодействие . . . 7
3.3 Тестирование . . . 7

A Глоссарий **8**

Список литературы **9**

Аннотация

Цель работы заключается реализации библиотеки построения симплициальных комплексов из облаков точек (например, комплекса Вьеториса — Рипса), их фильтраций, вычисления диаграмм устойчивости с помощью новейших алгоритмов редуцирования матриц инцидентности (например, алгоритм DoubleTwist), а также вычисления собственных векторов матрицы Лапласиана, чтобы идентифицировать расположение дырок в пространстве. Библиотека будет поддерживать два режима: самый быстрый и модульный для экспериментов.

Ключевые слова: топологический анализ, симплициальный комплекс, Вьеторис-Рипс, DoubleTwist, диаграмма устойчивости, параллельное вычисление, C++, оптимизация, Hodge Laplacian, persistent HH

1 Введение

Для ознакомления с основными понятиями см. *A*

Современный объём информации, которая нуждается в анализе, обработке и сравнении друг с другом, ставит новые вызовы для учёных и исследователей в области топологического анализа. Уже давно изучен базовый математический аппарат и имплементированы библиотеки, которые могут строить симплициальные комплексы и вычислять их диаграммы устойчивости, но они постепенно устаревают. На замену им приходят новые теории и алгоритмы, которые могут на порядок превосходить прежние алгоритмы по скорости.

Задача проекта заключается в создании библиотеки, которая реализует и комбинирует алгоритмы, собранные с множества новейших статей, для ускоренного и параллельного вычисления симплициальных комплексов, диаграмм устойчивости и собственных векторов матрицы Лапласиана. Библиотека будет иметь два направления:

1. Оптимизированная под конкретные алгоритмы для наибыстрейшего вычисления. Это позволит сравнить данную имплементацию с другими библиотеками на скорость
2. Модульная, с максимальным количеством шаблонов и абстракций. Данная версия необходима в научных и исследовательских целях для экспериментов над гипотезами. Она позволит быстро и удобно реализовывать нужные для исследований алгоритмы и объекты почти наравне с языком программирования Python

В поставленные задачи входит:

Изучение теории

Изучить теорию, стоящую за симплициальными комплексами, гомологическими группами, диаграммами устойчивости и гармоническими пространствами. Исследовать алгоритмы построения комплексов, оптимальные для их хранения структуры данных, фильтрацию, построение матрицы инцидентности, её редуцирование с целью получения диаграмм устойчивости, построения матрицы Лапласиана и вычисления собственных векторов.

Реализация необходимых объектов

Имплементировать на языке программирования C++ различные объекты для постройки нужного окружения для вычисления топологических объектов:

- Симплекс
- Симплициальный комплекс, который хранится в виде дерева симплексов [1]
- Фильтрация симплициального комплекса
- Столбец и матрица индексов симлпексов в отфильтрованном симплициальном комплексе
- Стоблец индексов для быстрого сложения
  1. В виде обычного вектора индексов
  2. В виде дерева битов [2]
- Диаграмма устойчивости
- Матрица Лапласиана [3]

Имплементация алгоритмов

Написать алгоритмы на полученных объектах:

- Быстрое построение комплекса Вьеториса-Рипса над облаком точек [4]
- Редуцирование матрицы инцидентности фильтрованного симплициального комплекса:
  1. Наивная реализация
  2. Параллельное редуцирование [5]
  3. Twist [6]
  4. DoubleTwist [7]
- Вычисление матрицы Лапласиана
- Вычисление собственных векторов с собственным значением 0

Разделение написанного кода

Разделить репозиторий на две ветки

1. В одной максимально абстрагировать полученные классы с целью получения лучшей среды для экспериментов
2. В другой рассмотреть разные комбинации алгоритмов и структур данных с целью выявить лучшую по скорости комбинацию и оптимизировать код под неё

Тестирование библиотеки

Провести стресс-тесты и тесты на корректность реализации алгоритмов. Измерить время работы для всех комбинаций алгоритмов и структур данных. Сделать выводы о лучшей компоновке. Для тестирования использовать готовое решение по тестированию проектов [8]

Изложение результатов

В отчёте изложить теорию, необходимую для понимания рассматриваемых алгоритмов и структур данных, а также описать архитектуру и дизайн библиотеки. Написать сопроводительную документацию для пользователей библиотеки.

2 Требования к проекту

2.1 Функциональные требования

Написанная библиотека на языке C++ должна предоставлять доступ к объектам, нужным для вычисления диаграмм устойчивости из облаков точек. Она должна обладать следующими объектами и методами:

2.1.1 Генерация комплекса Вьеториса-Рипса по облаку точек

Шаблонная функция, которая генерирует по массиву вершин и метрике комплекс Вьеториса-Рипса с заданными ограничениями по максимальному размеру симплексов и максимальной дистанции между вершинами в симплексе.

2.1.2 Симплекс

Шаблонный по типу вершины класс, содержащий вершины, которые принадлежат этому симплексу. Вершины должны быть упорядочены, то есть поддерживать `std::partial_ordering operator<=>`. Хранить их можно в векторе, сете или более хитрой структуре данных, например, в боре, где рёбра - номера вершин.

Класс симплексов должен предоставлять следующие публичные методы:

- Построение от итераторов на структуру, содержащую вершины

- Получение размерности симплекса = "количество вершин - 1". Симплексы без вершин создавать нет смысла, поэтому определение корректно
- Получение подсимплексов - симплексов размерности на 1 меньше, которые содержатся в нём
- Получение набора вершин симплекса, чтобы функциям фильтрации было с чем работать
- Строковое представление симплекса при выводе через `std::ostream& operator<<`, чтобы можно было выводить результат. Естественно, будет поддерживать только если и сами вершины можно вывести тем же способом

2.1.3 Симплициальный комплекс

Шаблонный по типу вершины класс, который должен содержать в себе симплексы на заданных вершинах. Наиболее оптимальная и корректная по определению структура для их хранения - это дерево симплексов.

Класс комплексов должен предоставлять следующие публичные методы:

- Построение от итераторов на структуру, содержащую симплексы
- Добавление произвольного числа симплексов
- Получение размера комплекса - количество содержащихся в нём симплексов
- Получение максимальной размерности симплексов в нём
- Получение набора симплексов в нём
- Строковое представление комплекса при выводе через `std::ostream& operator<<`, чтобы можно было выводить результат. Естественно, будет поддерживать только если и сами вершины можно вывести тем же способом

2.1.4 Фильтрация симплициального комплекса

Шаблонный по многим типам класс, который содержит в себе упорядоченное множество симплексов в комплексе, которые отсортированы по своим весам, которые определяются с помощью функции фильтрации, переданной в конструктор класса вместе с комплексом.

Класс фильтрации должен предоставлять следующие публичные методы:

- Построение от комплекса и функции фильтрации
- Получение упорядоченного набора симплексов
- Получение индексера, который по симплексу выдаёт его номер в этом упорядоченном наборе
- Получение максимальной размерности симплексов в нём

2.1.5 Столбцы и матрицы индексов

Основные классы хранения индексов симплексов, где индексы - номер симплекса в фильтрации. Наивная реализация - вектор и вектор векторов из индексов.

2.1.6 Столбцы для быстрого сложения

Классы, содержащие в своей структуре данных индексы. Наивная реализация - обёртка над вектором индексов.

Они должны поддерживать:

- Быстрое сложение с индексом и обычным столбцом из индексов, которые передаются в метод `void add` с помощью итераторов
- Нахождение максимального индекса, который содержится в нём
- Конвертация в обычный столбец индексов. То есть чтобы можно было быстро очистить столбец от содержащихся индексов и загрузить следующий

2.1.7 Диаграмма устойчивости

Шаблонный по типу вершины класс, который содержит в себе существенные и несущественные пары устойчивости.

Класс должен поддерживать:

- Добавление новой пары устойчивости по двум симплексам или одному, если это существенная пара
- Получение всех содержащихся пар
- Показ получившейся диаграммы с помощью вывода в `stdout` всех пар

2.1.8 Матрица Лапласиана

Класс, который содержит матрицу Лапласиана для некоторого момента времени редуцирования комплекса.

Он должен поддерживать:

- Создание из матрицы индексов и фильтрации симплициального комплекса
- Вычисление собственных векторов с собственным значением 0. Вектора имеют размерность количества вершин в комплексе.

2.1.9 Алгоритмы редуцирования

Шаблонные классы, которые строят матрицы инцидентности по фильтрации симплициального комлпекса и редуцируют её для получения диаграммы устойчивости. Планируется реализовать несколько вариантов, которые объявлены выше.

2.1.10 Алгоритм поиска persistent HH

Класс, который хранит в себе набор собственных векторов на текущий момент редуцирования, которые получает из матриц Лапласиана, и сравнивает их со следующим набором для поиска подходящего вектора, ответственного за создание некоторой дырки в комплексе.

По величинам значений элементов в этом векторе определяется расположение дырки в пространстве. [3]

2.2 Нефункциональные требования

2.2.1 Описание программы и используемых библиотек

1. Программа написана на языке C++, используется версия языка C++20
2. Для сборки и тестирования кода используется кросс-платформенная система сборки CMake
3. Программа поддерживает множество компиляторов, но основной - GCC
4. Проект использует систему поддержки версий git вместе с github
5. Требуется не менее 32GB RAM для корректной работы с большим объёмом данных
6. Для форматирования используется clang format
7. Используется clang-tidy для унификации именований и статической диагностики программы

2.2.2 Используемые библиотеки

1. Библиотека Boost.Asio для асинхронной и параллельной обработки матриц индексов [9]
2. Фреймворк GTest для тестирования программы [8]

2.2.3 Политика обработки ошибок

1. Обработка std::exception отключена в угоду оптимизации выполнения программы
2. Вставлены asserts в DEBUG версии программы для выявления ошибок имплементации методов и алгоритмов
3. Используется std::optional для обработки несуществования выходных данных алгоритмов и методов

Подытоживая, программа предполагает, что использующий её пользователь вводит верные данные и следует ограничениям используемых алгоритмов.

3 Теория и описание алгоритмов

В разработке

3.1 Дизайн решения

В разработке

3.2 Взаимодействие

В разработке

3.3 Тестирование

В разработке

A Глоссарий

Термины и обозначения:

- Вершины (V) - элементы линейно упорядоченного множества. В большинстве случаев это пронумерованные точки в пространстве $\mathbb{R}^3$
- Симплекс (σ) - фигура, объединяющая $n > 0$ вершин
- Размерность симплекса ($|\sigma|$) - количество вершин минус один
- k-симплекс (σ_k) - симплекс размерности k
- Подсимплекс (τ) симплекса (σ) - фигура, состоящая из $n - 1$ вершины симплекса. Обозначение: $\tau \in \sigma$
- Комплекс (K) - набор симплексов, который содержит все подсимплексы своих симплексов:

$$K \subset 2^V : \varnothing \in K;\ \ \forall \sigma \in K\ \ \forall \tau \in \sigma \implies \tau \in K$$

Список литературы

[1] Jean-Daniel Boissonnat and Clément Maria. The simplex tree: An efficient data structure for general simplicial complexes. *Algorithmica*, 70:406–427, 2014.

[2] Ulrich Bauer, Michael Kerber, Jan Reininghaus, and Hubert Wagner. Phat–persistent homology algorithms toolbox. *Journal of symbolic computation*, 78:83–84, 2017.

[3] Hyekyoung Lee, Moo K Chung, Hongyoon Choi, Hyejin Kang, Seunggyun Ha, Yu Kyeong Kim, and Dong Soo Lee. Harmonic holes as the submodules of brain network and network dissimilarity. In *Computational Topology in Image Context: 7th International Workshop, CTIC 2019, Málaga, Spain, January 24-25, 2019, Proceedings 7*, pages 110–122. Springer, 2019.

[4] Antonio Rieser. A note on the simplex-tree construction of the vietoris-rips complex. *arXiv preprint arXiv:2301.07191*, 2023.

[5] Erik von Brömssen. Computing persistent homology in parallel with a functional language. 2021.

[6] Chao Chen and Michael Kerber. Persistent homology computation with a twist. In *Proceedings 27th European workshop on computational geometry*, volume 11, pages 197–200, 2011.

[7] Tuyen Pham and Hubert Wagner. Computing representatives of persistent homology generators with a double twist. *arXiv preprint arXiv:2403.04100*, 2024.

[8] Google. Googletest. https://google.github.io/googletest/, 1 2025.

[9] Boost asio library. https://github.com/boostorg/asio, 1 2025.