АННОТАЦИЯ

В данном документе приведена пояснительная записка к программе "Программа для построения графов данных и поиска сообществ в них", предназначенной для построения графов данных и поиска сообществ в них.

В документе, в разделе "Введение" указано наименование программы, краткое наименование программы и документы, на основании которых ведется разработка.

В разделе "Назначение и область применения" указано функциональное назначение программы, эксплуатационное назначение программы и краткая характеристика области применения программы.

В разделе "Технические характеристики" содержатся следующие подразделы: постановка задачи на разработку программы, описание применяемых математических методов и алгоритмов, алгоритмы функционирования программы и ее интерфейс, организация входных и выходных данных, описание и обоснование выбора состава технических и программных средств.

В разделе "Ожидаемые технико-экономические показатели" указана предполагаемая потребность и полезность разработки, преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами.

Настоящий документ разработан в соответствии с требованиями:

1) ГОСТ 19.101-77 Виды программ и программных документов [1];

2) ГОСТ 19.102-77 Стадии разработки [2];

3) ГОСТ 19.103-77 Обозначения программ и программных документов [3];

4) ГОСТ 19.104-78 Основные надписи [4];

5) ГОСТ 19.105-78 Общие требования к программным документам [5];

6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [6];

7) ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению [7].

Изменения к данному Техническому заданию оформляются согласно ГОСТ 19.603-78 [8], ГОСТ 19.604-78 [9].

СОДЕРЖАНИЕ

1. Введение 5

1.1. Наименование Программы 5

1.2. Документ, на основе которого ведется разработка 5

2. Назначение и область применения 6

2.1. Назначение Программы 6

2.2. Краткая характеристика области применения 6

3. Технические характеристики 7

3.1. Постановка задачи на разработку программы 7

3.2. Применяемые математические методы 8

3.3. Алгоритмы функционирования программы и ее интерфейс 18

3.4. Организация входных и выходных данных 25

3.5. Выбор технических и программных средств 26

4. Ожидаемые технико-экономические показатели 27

5. Источники, использованные при разработке 28

Приложение 1. Таблицы с описанием классов и интерфейсов 30

Приложение 2. Таблицы с описанием методов 32

Лист регистрации изменений 42

**1. ВВЕДЕНИЕ**

**1.1. Наименование Программы**

Название программы: "Программа для построения графов данных и поиска сообществ в них". Изложено определение графов данных и сообществ, применяемых алгоритмов, описан и обоснован выбор технических средств, описан интерфейс программы и организация входных и выходных данных.

**1.2. Документ, на основе которого ведется разработка**

Приказ № 2.3-02/1212-02 от 12.12.2018 «Об утверждении тем и руководителей курсовых работ студентов 3 курса Факультета компьютерных наук, образовательная программа «Прикладная математика и информатика».

**Организация, утвердившая этот документ**: Национальный исследовательский университет «Высшая школа экономики», Факультет компьютерных наук, образовательная программа «Прикладная математика и информатика».

**Наименование темы разработки**: "Сложные сети: граф связей и граф данных».

**2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ**

**2.1. Назначение Программы**

Назначение разработки – построения графов данных и поиска сообществ в них.

**2.2. Краткая характеристика области применения**

Сообществами называются такие подмножества вершин, у которых много рёбер внутри подмножества и мало – с остальными вершинами[10, 11]. Существуют более строгие определения сообщества, но они не общеприняты. Поиск сообществ в сети важен, поскольку с его помощью можно изучить структуру сети, выявив в ней основные части и взаимодействия между ними.

Помимо рёбер на основе явно заданных связей между вершинами в сложной сети можно строить рёбра на основе метаданных, ассоциированных с вершинами, соединяя вершины со схожими метаданными. Такой граф называется графом данных (proximity graph). Существует несколько методов построения графа данных[12] (relative neigborhood graph, gabriel graph и т.д.), некоторые из которых планируется реализовать.

Таким образом, данная программа может применяться при исследовании различных наборов данных для выявления их структуры -- выделения основных частей данных и связей между ними.

Предполагается, что написанная библиотека будет использоваться для анализа данных, научных исследований и такого прочего.

**3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ**

**3.1. Постановка задачи на разработку программы**

Целью проекта является разработка и создание библиотеки на языке С++ для построения графов данных и поиска сообществ в них, а так же интерфейса на языке Python3 для этой библиотеки.

Реализованы функции построения следующих графов данных:

1. Граф k ближайших соседей.

Каждая вершина соединяется с её K ближайшими соседями.

2. Граф пар вершин на расстоянии меньше заданного.

При этом должна поддерживаться возможность не только явно задавать это расстояние, но и автоматически находить его, передавая в функцию желаемое количество ребёр в построенном графе.

3. RNG (relative neighbourhood graph)

Две точки в пространстве, соответствующие объектам из набора данных, соединяются рёбром, если в пересечении двух гиперсфер с центрами в каждой из точек и радиусом, равным расстоянию между ними, нет других точек из набора данных.

4. Граф Габриеля.

Две точки в пространстве соединяются рёбром, если в пересечении двух гиперсфер с центрами в каждой из точек и радиусом, равным расстоянию между ними, нет других точек из набора данных.

5. Граф сфер влияния.

Для каждой точки строится гиперсфера с радиусом, равным расстоянию до её ближайшего соседа. Далее пара точек соединяется ребром, если соответствующие им гиперсферы пересекаются.

Реализованы следующие алгоритмы поиска сообществ:

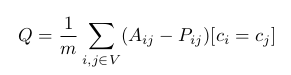
1. Label propagation, синхронная и асинхронная модификации.[13]

2. Алгоритм CNM.[14]

**3.2. Применяемые математические методы и алгоритмы программы**

Будем рассматривать задачу разбиения вершин графа на непересекающиеся подмножества – сообщества, отметив что помимо неё существует также задача разбиения на пересекающиеся сообщества (soft clustering).

Важнейший класс таких методов решения этой задачи основан на введении метрики качества разбиения вершин на сообщества и оптимизации введённого функционала. В качестве такого функционала часто берут модулярность, определяемую следующим образом:



Здесь m – количество ребёр, V – множество вершин, A – матрица смежности графа, ci – сообщество, к которому отнесена вершина i. Pij – некоторая оценка Aij для случайного графа, например (ki  kj ) / 2m , где ki – степень вершины i.

***3.2.1. CNM***

Задача оптимизации модулярности NP-полна, поэтому используемые на практике алгоритмы не ищут точное решение, а лишь стараются найти как можно лучшее приближение. Один из таких алгоритмов, CNM[5] – работает за O(n log2 n) для сложных сетей. Он создаёт по одному сообществу на каждую вершину, а затем на каждом шаге объединяет два таких сообщества, что при этом увеличение модулярности будет на данном шаге максимальным. Такие итерации продолжаются до тех пор, пока есть пара сообществ, объединение которых увеличит функцию модулярности. Таким образом, он жадно максимизирует функционал модулярности.

Процесс объединения сообществ можно изобразить лесом (графом, состоящим из нескольких деревьев), где вершины соответствуют сообществам, и потомки вершины соответствуют сообществам, объединением которых получилось соответствующее вершине сообщество. Такой граф называется дендрограммой. При использовании правильных структур данных данный алгоритм работает с временной сложностью O(n d log n), где n – размер графа, а d – высота дендрограммы. Поскольку для сложных сетей, которыми являются большинство встречающихся в реальной жизни графов, высота дендрограммы имеет порядок O(log n), сложность в таком случае получается равно O(n log2 n).

***3.2.2. Label propagation***

Другой подход к поиску сообществ основан на запуске случайного процесса, в результате работы получается разбиение множества вершин на сообщества. label propagation[6] – один из таких алгоритмов. Вначале каждой вершине присваивается своё сообщество, а затем на каждой итерации для каждой вершины считается число её соседей из каждого сообщества, и выбирается сообщество с максимальным числом соседей (если таких сообществ несколько, случайно выбранное из них). В зависимости от модификации либо сообщество вершины обновляется сразу после подсчёта (асинхронный вариант), либо сначала для всех вершин выбираются новые сообщества, а потом они одновременно обновляются после итерации по всем вершинам (синхронный вариант), либо сообщество вершины обновляется сразу, но вершины перебираются в таком порядке, что вершины из одного сообщества идут подряд (полу-синхронный вариант). Итерации продолжаются до тех пор, пока количество поменявших сообщество вершин на каждом шаге не станет меньше некоторого числа, выбираемого в зависимости от данных.

Каждая итерация имеет временную сложность O(n), при этом число итераций, необходимых, чтобы процесс сошёлся, невелико даже для больших n. Таким образом, данный алгоритм является масштабируемым.

**3.3. Алгоритмы функционирования программы и ее интерфейс**

В интерфейсе для языка C++ используются следующие определённые типы данных:

struct Edge {

int ind;

float w;

};

using Graph = vector<vector<Edge>>;

using LabelPropRes = std::pair<vector<int>, vector<int>>;

class FastMatrix;

Класс FastMatrix – класс для хранения двумерного массива вещественных чисел. В отличие от vector<vector<float>>, он хранит все данные в одном одномерном массиве, но при этом поддерживает удобную индексацию двумя индексами. Таким образом, в данной структуре данных не хранятся для каждой строки матрицы её размер вместимость и указатель на первый элемент. Кроме того, поскольку данные идут подряд, они лучше кешируются процессором.

Таблица с интерфейсом реализованных функций приведена в приложении.

**3.5. Выбор технических и программных средств**

Программа была разработана и проверена на функционирование под управлением операционной системы Linux, дистрибутива Ubuntu 18.04, архитектуры x86\_64, однако является кросплатформенной на стадии исходного кода и может использоваться на компьютере с любой операционной системой и архитектурой..

Исходными языками данной разработки являются языки C++ и Python3.

Для набора исходного кода использовался текстовый редактор vim.

В качестве системы контроля версий использовался git. Удалённый репозиторий хранился на сайте github.com.

Для использования библиотеки при программировании на языке С++ требуется компилятор языка С++, поддерживающий стандарт C++14. При использовании интерфейса библиотеки на языке Python3 требуется интерпретатор языка Python3.

**4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ**

Данная программа может применяться при исследовании различных наборов данных для выявления их структуры -- выделения основных частей данных и связей между ними.

В рамках данной работы расчёт экономической эффективности не предусмотрен. Продукт

будет распространяться бесплатно.

**5. ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ**

1. ГОСТ 19.101-77 Виды программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

2. ГОСТ 19.102-77 Стадии разработки. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

3. ГОСТ 19.103-77 Обозначения программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

4. ГОСТ 19.104-78 Основные надписи. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

5. ГОСТ 19.105-78 Общие требования к программным документам. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом.// Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

8. ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

9. ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

10. Santo Fortunato, Darko Hric, ”Community detection in networks: A user guide”, 2016.

11. Santo Fortunato, ”Community detection in graphs”, 2009.

12. Matthias Beck, ”Computational Discrete Geometry”

13. Usha Nandini Raghavan, Reka Albert, Soundar Kumara , ”Near linear time algorithm to detect community structures in large-scale networks”, 2009

14. Aaron Clauset, M. E. J. Newman,and Cristopher Moore, ”Finding community structure in very large networks”, 2004.

**ПРИЛОЖЕНИЕ 1**

**Таблицы с описанием функций**

**Таблица 1.** Интерфейс функций на языке С++

|  |  |  |  |
| --- | --- | --- | --- |
| имя | тип | аргументы | назначение |
| eps\_graph | vector<vector<int>> | const FastMatrix &data, int n\_edges | Построить граф пар вершин на расстоянии не более эпсилон. |
| eps\_graph\_n\_edges | vector<vector<int>> | const FastMatrix &data, int n\_edges | Построить граф пар вершин на расстоянии не более эпсилон, однако эпсилон подбирается автоматически так, чтобы в полученном графе было примерно n\_edges рёбер |
| knn\_graph | vector<vector<int>> | const FastMatrix &data, size\_t k | Построить граф k ближайших соседей |
| rng\_graph | vector<vector<int>> | const FastMatrix &data | Построить граф относительного соседства |
| gabriel\_graph | vector<vector<int>> | const FastMatrix &data | Построить граф Габриеля |
| influence\_graph | vector<vector<int>> | const FastMatrix &data | Построить граф сфер влияния |
| label\_propagation | LabelPropRes | const Graph &graph, int max\_iter, int min\_delta, int async=false | Поиск сообществ методом label propagation. Передаются условия остановки (максимальное количество итераций и минимальное количество замен меток на итерации), а так же аргумент, отвечающий за то, синхронная или асинхронная модификация алгоритма будет использоваться.  Возвращается пара из двух векторов. В первом – номер сообщества для каждой из вершин, а во второй количество изменений номеров на каждой итерации. |
| CNM | vector<int> | const Graph &graph | Поиск сообществ алгоритмом CNM. Возвращается вектор из номера сообщества для каждой из вершин. |

**Таблица 2** Интерфейс функций на языке Python3.

|  |  |  |
| --- | --- | --- |
| имя | аргументы | назначение |
| eps\_graph | data, epsilon | Построить граф |
| eps\_graph\_n\_edges | data, n\_edges | Построить граф пар вершин на расстоянии не более эпсилон, однако эпсилон подбирается автоматически так, чтобы в полученном графе было примерно n\_edges рёбер |
| knn\_graph | data, k | Построить граф k ближайших соседей |
| rng\_graph | data | Построить граф относительного соседства |
| gabriel\_graph | data | Построить граф Габриеля |
| influence\_graph | data | Построить граф сфер влияния |
| label\_propagation | graph, max\_iter, min\_delta, f\_async=false | Поиск сообществ методом label propagation. Передаются условия остановки (максимальное количество итераций и минимальное количество замен меток на итерации), а так же аргумент, отвечающий за то, синхронная или асинхронная модификация алгоритма будет использоваться.  Возвращается пара из двух векторов. В первом – номер сообщества для каждой из вершин, а во второй количество изменений номеров на каждой итерации. |
| CNM | graph | Поиск сообществ алгоритмом CNM. Возвращается вектор из номеров сообщества для каждой из вершин. |

**ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |