Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к магистерской диссертации

«Алгоритмы поиска подграфов социального графа, включающих заданное множество пользователей»

Автор: Филиппов Дмитрий Сергеевич _____
Направление подготовки (специальность): 01.04.02 Прикладная математика и информатика
Квалификация: Магистр
Руководитель: Фильченков А. А., канд. физ.-мат. наук _____

К защите допустить
Зав. кафедрой Васильев В.Н., докт. техн. наук, проф. _____

«__» _____ 20__ г.

Санкт-Петербург, 2019 г.

Студент Филиппов Д. С. **Группа** М4238 **Кафедра** компьютерных технологий **Факультет** информационных технологий и программирования

Направленность (профиль), специализация Технологии проектирования и разработки программного обеспечения

Квалификационная работа выполнена с оценкой _____

Дата защиты «21» июня 2019 г.

Секретарь ГЭК *Павлова О.Н.* Принято: «___» _____ 20___ г.

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

УТВЕРЖДАЮ
Зав. каф. компьютерных технологий
докт. техн. наук, проф.
_____ Васильев В.Н.
«___» _____ 20___ г.

ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ

Студент Филиппов Д. С. **Группа** М4238 **Кафедра** компьютерных технологий **Факультет** информационных технологий и программирования **Руководитель** Фильченков А. А., канд. физ.-мат. наук, доцент ФИТиП

1 Наименование темы: Алгоритмы поиска подграфов социального графа, включающих заданное множество пользователей

Направление подготовки (специальность): 01.04.02 Прикладная математика и информатика

Направленность (профиль): Технологии проектирования и разработки программного обеспечения

Квалификация: Магистр

2 Срок сдачи студентом законченной работы: «31» мая 2019 г.

3 Техническое задание и исходные данные к работе.

По данному неориентированному невзвешенному графу социальной сети и выделенным в нем вершинам-запросам требуется найти плотный подграф, содержащий все или большинство данных вершин, то есть так называемое сообщество социальной сети.

4 Содержание магистерской диссертации (перечень подлежащих разработке вопросов)

Пояснительная записка должна демонстрировать новый подход к решению этой задачи, а также его плюсы и минусы по сравнению с предыдущими методами. Должно быть произведено сравнение оптимальности ответов, времени работы, а также других метрик для всех рассмотренных и приведенных алгоритмов.

5 Перечень графического материала (с указанием обязательного материала)

Не предусмотрено

6 Исходные материалы и пособия

а) Faloutsos C., McCurley K.S. and Tomkins A. Fast discovery of connection subgraphs;
б) Faloutsos C., Tong H. Center-Piece Subgraphs: Problem Definition and Fast Solutions;
в) Ruchansky N. et al. The Minimum Wiener Connector Problem;
г) Gionis A., Mathioudakis M. and Ukkonen A. Bump hunting in the dark: Local discrepancy maximization on graphs.

7 Календарный план

№№ пп.	Наименование этапов магистерской диссертации	Срок выполнения этапов работы	Отметка о выполнении, подпись руков.
1	Ознакомление с исходными статьями	10.2016	
2	Поиск новых статей	11.2016	
3	Изучение новых статей, выбор бейзлайна	12.2016	
4	Реализация бейзлайна	03.2017	
5	Исследование темы, предложения улучшений бейзлайна	03.2017	
6	Реализация улучшений, сравнение практических результатов с теоретическими	04.2017	
7	Написание пояснительной записки	05.2017	

8 Дата выдачи задания: «01» сентября 2017 г.

Руководитель _____

Задание принял к исполнению _____ «01» сентября 2017 г.

Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

АННОТАЦИЯ
МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

Студент: Филиппов Дмитрий Сергеевич

Наименование темы работы: Алгоритмы поиска подграфов социального графа, включающих заданное множество пользователей

Наименование организации, где выполнена работа: Университет ИТМО

ХАРАКТЕРИСТИКА МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

1 Цель исследования: В рамках работы необходить предложить улучшение существующих методов поиска сообщества по выделенным в графе социальной сети вершинам, предложить алгоритм отсеивания шума в запросе для получения более плотного подграфа в качестве ответа.

2 Задачи, решаемые в работе:

а) Провести исследование описанной задачи;
б) Выделить одно или несколько базовых решений;
в) Реализовать выбранные базовые решения и сравнить их результаты с результатами с статьях;
г) Разработать методы улучшения базовых решений, способные учитывать шум в запросах;
д) Реализовать разработанные методы и сравнить полученные результаты с результатами базовых решений.

3 Число источников, использованных при составлении обзора: 15

4 Полное число источников, использованных в работе: 15

5 В том числе источников по годам

Отечественных			Иностранных		
Последние 5 лет	От 5 до 10 лет	Более 10 лет	Последние 5 лет	От 5 до 10 лет	Более 10 лет
6	5	4	0	0	0

6 Использование информационных ресурсов Internet: нет

7 Использование современных пакетов компьютерных программ и технологий: Для реализации алгоритмов был использован язык программирования *Java 1.8*. Также был использован фреймворк *Kryo* для быстрой и автоматической сериализации и десериализации большого объема данных. Для вычислений, использующих большой объем памяти, были использованы технологии *ssh* и *slurm* для подключения и работы на серверах кластера Университа ИТМО, имеющих 128, 256 и 496 Гб оперативной памяти.

8 Краткая характеристика полученных результатов: Результаты, полученные в статье, показывают, что на запросах, содержащих шум, предложенное решение работает оптимальнее всех предыдущих. На запросах без шума решение работает не хуже, а иногда даже лучше существующих решений.

9 Гранты, полученные при выполнении работы:

10 Наличие публикаций и выступлений на конференциях по теме работы:

Выпускник: Филиппов Д. С. _____

Руководитель: Фильченков А. А. _____

«__» _____ 20__ г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Последнее время изучение и аналих социальных сетей представляет большой интерес. Один из примеров анализа социальных сетей — нахождение сообществ пользователей. Многие работы исследуют сообщества целого графа, в то время как большой интерес также представляет анализ сообществ, образованных только данным множеством вершин — по данным выделенным вершинам в социальном графе найти плотный подграф, содержащий их все. Эта задача также широко изучена, однако наложение условия на наличие всех вершин в итоговом подграфе не всегда оптимально для нахождения наиболее плотного подграфа, так как этот подход не учитывает возможный «шум» в запросе.

В этой работе представлен метод оптимизации текущих методов решения описанной задачи при условии возможного шума в запросе. Метод показал хорошие результаты на реальных графах социальных сетей, а также улучшил результаты предыдущих работ.

Актуальность исходной задачи (требующей наличие всех выделенных вершин в итоговом подграфе) проявляется во многих областях:

а) Полиция — зная нескольких подозреваемых, выяснить, кто еще мог быть соучастником преступления, участником банды или группировки;
б) Социальные сети — после добавления одного или нескольких друзей в социальной сети, также предлагается еще несколько, которые тесно связаны с недавно добавленными и которых вы вероятно всего тоже знаете;
в) Медицина — по нескольким заболевшим определить других наиболее вероятно инфицированных, используя социальный граф связей и знакомств;
г) Организация мероприятий — если на важное мероприятие требуется позвать несколько спикеров, также понять, кого еще, тесно связанного с этими людьми, хорошо было бы увидеть на этом мероприятии.

Актуальность нашей задачи (требующей наличие не всех, а только большинства выделенных вершин в итоговом подграфе) также проявлется в этих областях, причем, как можно заметить, наша задача более актуальна в реальной жизни:

а) Полиция — определить остальных соучастников преступления, участников банды или группировки, учитывая, что некоторые подозреваемые могли быть взяты только для этого дела и к группировке отношения не имеют;

б) Социальные сети — рекомендация друзей после недавнего добавления нескольких людей, учитывая то, что добавленные друзья могут быть из разных социальных сообществ и связь между ними может быть только через вас;
в) Медицина — определить наиболее вероятно инфицированных, учитывая тот фактор, что некоторые результаты могли оказаться ложноположительными;
г) Организация мероприятий — приглашение людей, наиболее связанных со спикерами, учитывая то, что спикеры могут быть слабо связаны друг с другом.

Практически все существующие по этой теме статьи рассматривают задачу поиска подграфа, содержащего все выделенные вершины. Наш же алгоритм вводит возможность взятия в итоговое подмножество не всех выделенных вершин, а только большинства из них, и показывает результаты лучше, чем предыдущие алгоритмы на запросах, содержащих шум.

В главе 1 приведен обзор темы: введены основные определения, используемые в статье и необходимые для понимания остальных терминов, приведен обзор существующих решений поставленной задачи с кратким описанием каждого решения. В конце главы уточнены требования к работе исходя из изложенной информации.

В главе 2 предложено подробное описание алгоритма, разбитое на фазы. В главе также приведены примеры работы алгоритма на небольших запросах с подробным описанием и иллюстрациями.

В главе 3 приведено описание проведенных экспериментов на реальных данных — какие данные были выбраны для анализа, как строились и проводились эксперименты. Также приведены диаграммы и таблицы, описывающие результаты экспериментов, показывающие и доказывающие улучшения, описанные в этой работе.

ГЛАВА 1. ОБЗОР ПОНЯТИЙ И СУЩЕСТВУЮЩИХ РЕШЕНИЙ

В данный момент проблема автоматизации различных вещей, таких как поиск сообществ в социальных сетях, сильно актуальна и активно развивается. На данный момент разразботаны различные методики выделения сообществ во всем графе социальной сети [1–4], а также выделения плотного сообщества, содержащего все выделенные вершины в графе [5–8]. Также были предложены методы разбиения вершин в запросе на несколько сообществ [9] — алгоритм DOT2DOT. Однако, почти все эти методики не поддерживают шум в запросе и выдают неоптимальные подграфы на такие запросы. Именно эту проблему мы решаем в этой работе, предложив модификации существующих алгоритмов, позволяющие эффективно выделять искомые подграфы даже если в исходном запросе существует некий шум.

1.1. Термины и понятия

В данном разделе описаны основные термины, используемые в других частях представленной работы.

1.1.1. Вводные понятия теории графов

Декартовым произведением $A \times B$ двух множеств A и B называется множество всех пар (a, b), таких, что $a \in A, b \in B$.

Граф — абстрактный математический объект, представляющий из себя множество *вершин*, некоторые пары из которых соединены *ребрами*. Здесь и далее, говоря «`граф`», мы будем иметь в виду «`неориентированный граф`», то есть граф, ребра которого неориентированы. Более формально, **граф** или **неориентированный граф** G — это упорядоченная пара $G := (V, E)$, где V — непустое множество вершин или узлов графа G, а E — множество неупорядоченных пар вершин, называемых *ребрами* ($E \subset V \times V$).

Через **V(G)** будем обозначать множество вершин графа G, а через **E(G)** множество ребер графа G. Через **|V(G)|** будем обозначать количество вершин графа G, а через **|E(G)|** количество ребер графа G. Если в контексте понятно, о каком графе идет речь, будем просто писать $|V|$ и $|E|$ соответственно.

Через $N_G(v)$ будем обозначать множество соседей вершины v в графе G, то есть множество вершин, напрямую соединенных с v ребром: $N_G(v) = \{u | (v, u) \in E(G)\}$.

Степенью вершины v графа G называется количество ребер, исходящих из этой вершины, т.е. $deg(v) = |N_G(v)|$.

Подграфом графа G называется граф $G' := (V', E')$, такой, что $V' \subseteq V(G)$ и $E' \subseteq E(G) \cap (V' \times V')$. Иными словами, это граф, порожденный подмножеством вершин исходного графа и содержащий только ребра исходного графа G между вершинами этого подмножества.

$G[V]$ называется **порожденным подграфом** графа G множеством вершин V, если $G[V] := (V, E[G, V])$, где $E[G, V]$ — подмножество ребер графа G, оба конца которых содержатся в V, то есть $E[G, V] = E(G) \cap (V \times V)$.

k-truss графа G называется максимальным по количеству вершин подграф $G' \subseteq G$, такой, что для каждого его ребра (v, u) количество вершин w, таких, что, в G' существуют ребра (w, v) и (w, u), не меньше k. Другими словами, $k - truss$ — наибольший по размеру подграф G', для каждого ребра (v, u) которого $|N_{G'}(v) \cap N_{G'}(u)| >= k$.

Простым путем графа G называется такой набор его вершин $v_1, v_2, v_3, \ldots, v_k$, что $\forall\ i, j : v_i \neq v_j$, а также $\forall\ 1 \leqslant i \leqslant k - 1 : (v_{i-1}, v_i) \in E(G)$.

Кратчайшим путем из вершины v в вершину u называется простой путь $v_1, v_2, \ldots, v_n$ минимальной длины, то есть с минимальным n. Длина кратчайшего пути из v в u в графе G обозначается как $d_G(v, u)$.

Диаметром графа G называется длина максимального по длине кратчайшего пути в G. Другими словами, $diam(G) = max_{v,u \in V(G)}\ d_G(v, u)$.

k-core графа G называется максимальный по количеству вершин подграф $G' \subseteq G$, такой, что степень каждой его вершины не меньше k. Для фиксированного k, через C_k будем обозначать *k-core*, а именно набор компонент связности, из которых он состоит. Таким образом, $C_k = \{H_i\}$, где H_i — i-я компонента связности, в которой степень всех вершин хотя бы k. Число k будем называть **порядком** *k-core*.

Через $\boldsymbol{\mu(G)}$будем обозначать минимальную степень вершин в графе G, т.е. $\mu(G) = \min_{v \in V(G)} deg(v)$.

Core decomposition или **ядерной декомпозицией** будем называть набор *k-core* для всех возможных k: $C = \{C_k\}_{k=1}^{k=k^*}$. Стоит отметить, что из определения *k-core* следует, что $C_1 \supseteq C_2 \supseteq C_3 \ldots \supseteq C_{k^*}$.

Core index или **ядерным индексом** вершины v будем называть минимальный по размеру *k-core*, содержащий v, т.е. *k-core* с максимальным k: $c(v) = \max(k \in [0..k^*] | v \in C_k)$.

γ-quasi-clique графа G называется любой такой подграф $G' \subseteq G$, что он «достаточно плотный», а именно: $\frac{2 \cdot |E(G')|}{|V(G')| \cdot (|V(G')| - 1)} \geqslant \gamma$.

1.1.2. Социальные сети

Сообществом или **Сообществом социальной сети** называется набор вершин графа социальной сети G, где все вершины объединяет некоторое свойство или признак. Например, «сообщество любителей рока» или «сообщество акционеров Газпром».

Социальной кликой или **кликой** будем называть множество людей в социальной сети, где каждый человек знает всех остальных в этой клике, другими словами, между любой парой различных вершин в этой клике есть ребро.

Социальной псевдокликой или **псевдокликой** будем называть множество людей, где необязательно каждая пара различных вершин соединена ребром, однако множество людей все равно достаточно плотно связано. Оценка, насколько хорошо много связана будет зависеть от выбора типа псевдоклики и будет описана дальше в работе, однако во всех описаниях основную роль играет количество ребер в подграфе по отношению к количеству пар вершин ($\frac{2 \cdot |E(G)|}{|V(G)| \cdot (|V(G)| - 1)}$).

Free-rider effect называется эффект, возникающий при получении ответа на сформулированную задачу (поиск плотного сообщества по набору вершин), который содержит в себе ненужные подграфы — подграфы, которые можно удалить без нарушения оптимальности ответа, тем самым, уменьшив размер итогового подграфа, что является одной из первоочередных целей нашей задачи.

1.1.3. Используемые сокращения

RW — Random Walks. Идея основана на перемещении из текущей вершины в соседнюю по ребру с вероятностью, пропорциональной весу ребра.

RWR — Random Walks with Restarts. Идея аналогична идее RW, однако в этом случае также существует вероятность пойти из текущей вершины в исходную, в которой все было начато.

Smart-ST — Smart Spanning Trees. Эвристика для решения задачи Штейнера, а также используемая в статье Gionis et al. [10].

CSP — Community Search Problem. Это задача нахождения сообщества в социальной сети, содержащего все выделенные вершины.

NCSP — Noising Community Search Problem. Это задача нахождения сообщества в социальной сети, содержащего большинство выделенных вершины (то есть отсеивая шум).

1.2. Обзор существующих решений

Задача, рассматриваемая в этой статье, формулируется следующим образом: по данному графу G и набору выделенных вершин $Q \subset V(G)$ требуется решить задачу поиска сообщества, содержащего большинство, но не обязательно все вершины из Q. Выделенные вершины из множества Q мы также иногда будем называть «вершинами-запросами», «запросом» или «вершинами из запроса».

1.2.1. Исходные решения

а) Задача поиска сообщества по выделенным вершинам рассматривается уже довольно давно. Еще в 2004 году Faloutsos et al. [11] предложили алгоритм нахождения плотного сообщества по двум выделенным в графе вершинам ($|Q| = 2$). В алгоритме показывается неоптимальность метрик плотности «кратчайший путь» и «максимальный поток» между двумя заданными вершинами. Вместо этого, исходный граф представляется в виде электрической сети и используется метрика «доставленный ток между вершинами» — устанавливая напряжение `+1` на первую вершину-запрос и `0` на вторую, находится подграф, который доставляет наибольший ток между вершинами из запроса. Приведенная метрика подходит только для $|Q| = 2$, однако этот алгоритм положил начало изучению этой темы. В дальнейшем многие авторы статей оптимизировали результаты этой статьи.

б) Авторы второй статьи, рассмотренной для изначального изучения [5], предлагают функцию метрики плотности на основе *random walks with restarts* (RWR), дословно переводящееся как *случайные прогулки с возвращениями* (однако мы не будем использовать этот термин), используемую на взвешенном графе. Они рассматривают $r(i,j)$ — вероятность

того, что начав в i-й вершине-запросе q_i, с помощью RWR, где на каждом шаге из текущей вершины мы переходим в соседнюю по ребру с вероятностью пропорциональной весу ребра, мы закончим в вершине j. Также вводится $r(Q, j)$, равная сумме $r(i, j)$ для всех вершин-запросов: $r(Q, j) = \sum_{i=1}^{i=|Q|} r(i, j)$. Метрика плотности вводится как $g(H) = \sum_{j \in H} r(Q, j)$. Этот метод показал достаточно хорошие результаты по сравнению со статьей 2004 года, так как расширил возможность поиска подграфа с $|Q| = 2$ до $2 \leqslant |Q| \leqslant |V(G)|$, а также ввел возможность поиска подграфа, содержащего не все вершины, а только хотя бы k из них (k — параметр, данный на входе). Это операция была названа $K\_softAND$ и была успешно реализована в статье. Последующие алгоритмы развивали эту тему, применяли другие метрики и улучшили результаты этого алгоритма, однако задача поиска подграфа, содержащего не обязательно все, а только часть вершин-запросов в ответе, больше практически не поднималась, улучшения операции $K\_softAND$ не рассматривались ввиду количества других возможных применений и улучшений поставленной задачи.

в) Авторы третьей статьи, рассмотренной для изначального изучения [6], предлагают в качестве метрики плотности брать *индекс Винера* — попарную сумму кратчайших расстояний между вершинами из запроса. Авторы статьи пытаются решить проблему большого графа в результате обрабатывания запроса, если вершины-запросы находятся в нескольких сообществах и слабо связаны между собой. Для решения этой проблемы авторы предлагают добавлять в запрос несколько «важных вершин», который будут связывать сообщества, пусть и не очень плотно. Результаты показали, что этот метод работает в несколько раз лучше большинства предыдущих [5, 12] и примерно так же, как методы, основанные на *проблеме Штейнера*. Однако в статье не рассмотрены более поздние методы на основе *проблемы Штейнера*, которые заметно улучшили результаты предыдущих, что делает этот метод менее приоритетным по сравнению с ними.

1.2.2. Нахождение оптимальных псевдоклик

Большу́ю часть всех алгоритмов для решения описанной задачи занимают алгоритмы, основанные на нахождении оптимальных псевдоклик с некоторыми дополнительными эвристиками. Рассмотренных в алгоритмах псевдоклик до-

вольно много: например, *k-core* [8], *k-truss* [7], γ*-quasi-clique* [13] или просто алгоритмы, максимизирующие плотность ребер в полученном подграфе [14], что, фактически, и является определением псевдоклики. Для каждой из этой псевдоклик алгоритмы развиваются, улучшают полученные результаты, используя новые эвристики. Сравнивать результаты алгоритмов, использующих разные псевдоклики, довольно сложно и вряд ли даст видимые результаты, потому что из-за разницы оптимизируемых функций полученные результаты сильно зависят от исходных графов и запросов. В некоторых случаях определенные псевдоклики будут давать результаты лучше других, в некоторых — хуже, поэтому имеет смысл сравнивать только средние показатели результатов, однако это тоже не дает полного понимания оптимальности или неоптимальности алгоритмов.

Рассмотрим несколько последних алгоритмов для наиболее популярных псевдоклик:

a) X. Huang et al. [7] в качестве псевдоклик выбирают *k-truss*. Однако, просто нахождение оптимального *k-truss* (то есть *k-truss*, содержащий в себе все вершины-запросы, с максимальным k) — не оптимально, а также это уже решенная за полиномимальное время задача. Поэтому авторы статьи предлагают находить *k-truss* с максимальным k и минимальным диаметром, что, как они показывают в своей статье, уже NP-полная задача. Однако, эта идея должна показывать хорошие результаты, поэтому авторами была проведено исследование в попытке понять, насколько близкий ответ можно получить за полиномиальное время. Оказалось, что задачу нельзя решить с точностью лучше, чем в $(2 - \varepsilon)$ раз хуже для любого $\varepsilon > 0$ (под точностью подразумевается длина диаметра в полученном ответе). Однако, авторами был предложен эвристический алгоритм, который в худшем случае делает ошибку ровно в 2 раза, что показывает, что он является оптимальным для решения поставленной авторами задачи. Алгоритм основывается на построении предполагаемого максимального *k-truss* с последующий итеративным удалением вершин, не ухудшающих ответ и уменьшающих длину диаметра. Результаты, полученные в статье также являются довольно неплохими по сравнению с предыдущими статьями [12, 14], однако, даже несмотря на то, что в статье доказана полная оптимальность разработанного алгоритма, оптимальным для решения всей задачи

(нахождения плотного подмножества по заданному множеству вершин) он не является, поскольку авторами был разработан алгоритм только для поставленной *ими* задачи.

б) N. Barbieri et al. [8] в качестве псевдоклик выбирают *k-core*. Однако, и здесь простое нахождение оптимального *k-core* (то есть *k-core*, содержащий в себе все вершины-запросы, с максимальным k) — не оптимально, а также уже решено за полиномиальное время [12]. Поэтому авторы статьи применяют эвристики, нацеленные на минимизацию размера итогового подграфа с сохранением его оптимальности. Эти эвристики позволяют свести задачу к поиску ответа в компоненте связности $H^* \subset G$, причем при этом гарантируется, что все возможные оптимальные ответы на исходную задачу лежат в H^*. После этого авторы статьи приводят несколько эвристик для минимизации полученного подграфа H^*, содержащего все решения задачи. Само утверждение, описанное авторами, не ново, однако выглядит интересным для дальнейших исследований, потому что добавляет в постановку задачи больше информации без потери решений. Результаты статьи показывают, что метод действительно работает лучше и быстрее предыдущих [12, 15], причем результаты улучшены примерно так же хорошо, как и у X. Huang et al. [7]. Исходя из информации, описанной выше, было сделано решение выбрать эту статью как основу для улучшений и расширений на нашу постановку задачи.

в) Y. Wu et al. [14] в качестве псевдоклик выбирают *query-biased* плотность ребер, основанную на *RW*. Авторы нацелены на устранение *free-rider effect*, который проявляется во многих имеющихся алгоритмах, поэтому формулируют новую задачу поиска подграфа, содержащего все вершины-запросы с максимальной *query-biased* плотностью. Результаты статьи показывают, что метод действительно оптимизует существующие решения и практически не проявляет в себе *free-rider effect*. Он показал результаты лучше существовавших на тот момент *k-core* решений [12], γ*-quasi-clique* решений [13], а также некоторых других решений. Однако, новые решения [7, 8] также практически лишены *free-rider effect*, поэтому по сравнению с ними этот алгоритм будет не оптимальнее.

1.2.3. Другие методики

Как уже было рассмотрено ранее, оптимизируемые функции бывают довольно разные. В предыдущей части были рассмотрены псевдоклики, здесь мы рассмотрим несколько других популярных оптимизируемых функций.

а) L. Akoglu et al. [9] немного меняют постановку задачи, пытаясь найти подграф, объединяющий не все вершины в запросе, а различные их группы. То есть идея фактически состоит в разбиении запроса на группы и построения ответа для каждой группы отдельно, чтобы в каждой группе вершины были плотно связаны и объединены каким-то общим свойством, а между собой группы были связаны не очень сильно. Это соответствует разбиению запроса на несколько сообществ. Результаты статьи показали, что этот метод довольно неплохо решает поставленную в статье задачу, однако, как уже было сказано выше, эта задача отличается от нашей и сравниваться с ней довольно сложно. Ее можно свести к нашей задаче, однако это сведение не равносильное и требует дополнительного исследования.

б) A. Gionis et al. [10] в своей статье рассматривают метрику *линейное локальное несоответствие* (она же *linear local discrepancy*), которая равна взвешенной разнице количества вершин-запросов в итоговом подграфе-ответе и количества остальных вершин. Более точно, $g(C) = \alpha p_C - n_C$, где $p_C = |Q \cap V(C)|$, а $n_C = |V(C) \setminus Q|$. Алгоритм, максимизирующий эту функцию, основан на *проблеме Штейнера*, описанной ранее, а также *«Умных» минимальных остовных деревьяв* (они же *Smart-ST*). Отличительной чертой этого алгоритма является возможность решать задачу, используя *локальную модель доступа*, то есть модель, где весь граф изначально неизвестен (или он слишком большой, чтобы его полностью и быстро сохранить в оперативную память или на диск), и API позволяет только делать запросы на получение всех соседей вершины — $get - neighbours$. Эта модель позволяет решать задачу оптимально даже на очень больших графах социальных сетей, таких как *Twitter*, *Facebook*, а также многих других. Исходя из постановки задачи, в итоговом найденном подграфе вполне могут содержаться не все вершины, что совпадает с темой нашего исследования, однако метрика не очень подходит именно к нашей задаче — в ней не учитывается реберная плотность полученного подграфа, а учитывается толь-

ко соотношение количества вершин. Также в ответ вполне может войти довольно мало вершин из исходного запроса, нас такое тоже не устраивает.

1.3. Уточненные требования к работе

Подведем итог описанного выше обзора имеющихся решений. Большинство текущих рещений достаточно оптимально решают CSP — каждый использует свою метрику, но получает достаточно хорошие результаты. Однако, как мы можем заметить, что решение NCSP, то есть учитывание «шума», почти не поднимается в текущих решениях. Мы отметили рассмотрение задачи NCSP в статьях C. Faloutsos & H. Tong [5], а также A. Gionis et al. [10], однако там эта задача не является основной и цель сфокусировать результаты на этом не является первоочередной. Соответственно, целью нашей статьи будет разработка алгоритма, фокусирующегося на поиске плотного подграфа, не содержащего в себе исходной шум в запросе. Требования к нему будут следующие:

а) Алгоритм должен показывать результаты лучше текущих имеющихся решений [5, 8, 10];
б) Алгоритм должен быть достаточно оптимальным, желательно не проигрывающим по времени работы текущим аналогам;
в) Плюсом будет поддержка обратной совместимости — есть пользователь захочет все-таки найти подграф, содержащий все вершины в запросе, это должно быть возможно;

Выводы по главе 1

В главе были описаны основные определения, требуемые для понимания статьи и ее терминов, а также приведен обзор текущих решений поставленной задачи. Обзор показал, что за 13 лет было изучено множество подходов к решению CSP, однако изучаемая в этой бакалаврской работе NCSP изучена не очень хорошо, поэтому ее усовершенствование более чем резонно.

В конце главы были приведены новые требования к работе, мотивирующие и показывающие возможность создания алгоритма, работающего лучше предыдущих исследований.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Newman M.* Fast algorithm for detecting community structure in networks. — 2004.

2 *Newman M.* Finding community structure in networks using the eigenvectors of matrices // Phys. Rev. E 74. — 2006. — Т. 74.

3 *Fortunato S.* Community detection in graphs. — 2010.

4 Online search of overlapping communities / W. Cui [и др.] // Proceeding SIGMOD '13 Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. — 2013. — Т. 978. — С. 277–288.

5 *Faloutsos C.*, *Tong H.* Center-Piece Subgraphs: Problem Definition and Fast Solutions // Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and Data Mining. — 2006. — Т. 1. — С. 404–413.

6 The Minimum Wiener Connector Problem / N. Ruchansky [и др.] // Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. — 2015. — Т. 1. — С. 1587–1602.

7 Approximate closest community search in networks / X. Huang [и др.] // Proceedings of the VLDB Endowment. — 2015. — Т. 9. — С. 276–287.

8 Efficient and effective community search / N. Barbieri [и др.] // Data Mining and Knowledge Discovery. — 2015. — Т. 29. — С. 1406–1433.

9 Mining Connection Pathways for Marked Nodes in Large Graphs / L. Akoglu [и др.]. — 2013.

10 *Gionis A.*, *Mathioudakis M.*, *Ukkonen A.* Bump hunting in the dark: Local discrepancy maximization on graphs // 2015 IEEE 31st International Conference on Data Engineering (ICDE). — 2015. — Т. 978. — С. 1155–1166.

11 *Faloutsos C.*, *McCurley K. S.*, *Tomkins A.* Fast discovery of connection subgraphs // Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. — 2004. — Т. 1. — С. 118–127.

12 *Sozio M.*, *Gionis A.* The community-search problem and how to plan a successful cocktail party // Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. — 2010. — Т. 978. — С. 939–948.

13 *Zhu L.*, *Ng W. K.*, *Cheng J.* Structure and attribute index for approximate graph matching in large graphs // Journal Information Systems. — 2011. — Т. 36. — С. 958–972.

14 Robust local community detection: on free rider effect and its elimination / Y. Wu [и др.] // Journal Proceedings of the VLDB Endowment. — 2015. — Т. 8. — С. 798–809.

15 Local search of communities in large graphs / W. Cui [и др.] // Proceeding SIGMOD ’14 Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. — 2014. — Т. 978. — С. 991–1002.