Омский филиал Федерального государственного бюджетного учреждения
науки Институт математики им. С.Л. Соболева Сибирского отделения
Российской академии наук

На правах рукописи

МОРШИНИН Александр Владимирович

ПРИБЛИЖЕННОЕ И ТОЧНОЕ РЕШЕНИЕ РАЗЛИЧНЫХ ВАРИАНТОВ ЗАДАЧИ КЛАСТЕРИЗАЦИИ ВЕРШИН ГРАФА

Специальность 01.01.09 – дискретная математика и
математическая кибернетика

Диссертация на соискание ученой степени
кандидата физико-математических наук

Научный руководитель
д.ф.-м.н., профессор
В.П. Ильев

Омск – 2022

Оглавление

Введение **4**

1 Задача кластеризации вершин графа с ограниченным числом кластеров **17**

1.1 Известные результаты для задачи $\mathbf{GC_{\leqslant 2}}$ 17

1.1.1 Постановка задачи и вспомогательные утверждения . . . 17

1.1.2 Приближенные алгоритмы для задачи $\mathbf{GC_{\leqslant 2}}$ 18

1.2 Приближенные алгоритмы для задачи $\mathbf{GC_{\leqslant 3}}$ 22

1.2.1 6-приближенный алгоритм для задачи $\mathbf{GC_{\leqslant 3}}$ 22

1.2.2 Приближенный алгоритм для задачи $\mathbf{GC_{\leqslant 3}}$, не использующий локальный поиск 26

2 Задачи кластеризации вершин графа с фиксированным числом кластеров **33**

2.1 Задача $\mathbf{GC_2}$ 33

2.1.1 Постановка задачи и вспомогательные утверждения . . . 34

2.1.2 3-приближенный алгоритм для задачи $\mathbf{GC_2}$ 35

2.1.3 Процедура локального поиска 38

2.1.4 2-приближенный алгоритм для задачи $\mathbf{GC_2}$ 39

2.2 Задачи и методы кластеризации с частичным обучением 48

2.2.1 Постановки задач и вспомогательные утверждения . . . 48

2.2.2 Приближенные алгоритмы для задач $\mathbf{SGC_2}$ и $\mathbf{SSGC_2}$. 49

2.3 Взаимосвязь задач $\mathbf{GC_2}$, $\mathbf{SGC_2}$ и $\mathbf{SSGC_2}$ 53

3 Точные методы и экспериментальное исследование задач кластеризации вершин графа **55**

3.1 Точные методы для задач кластеризации вершин графа 55
3.1.1 Постановки задач и вспомогательные утверждения . . . 56
3.1.2 Метод ветвей и границ 57
3.1.3 Модели целочисленного линейного программирования . . 63
3.2 Экспериментальное исследование приближенных и точных алгоритмов для задач кластеризации вершин графа 67
3.2.1 Описание вычислительного эксперимента 67
3.2.2 Экспериментальное исследование алгоритмов для задач $\mathbf{GC}_{\leqslant 2}$, $\mathbf{GC}_{2}$, $\mathbf{SGC}_{2}$ и $\mathbf{GC}_{\leqslant 3}$ 70
3.2.3 Общий вывод по результатам экспериментального исследования 86

Заключение **88**

Литература **89**

Публикации автора по теме диссертации **94**

Приложения **96**

Введение

В диссертационной работе исследуются различные варианты задач кластеризации вершин графа. В задачах кластеризации требуется разбить заданное множество объектов на несколько подмножеств (кластеров) так, чтобы минимизировать число пар похожих объектов, попавших в разные кластеры, плюс число пар непохожих объектов, оказавшихся в одном кластере. В задаче кластеризации вершин графа отношение сходства объектов задается при помощи ребер неориентированного графа, вершины которого взаимно однозначно соответствуют объектам. В машинном обучении задачи кластеризации относят к разделу обучения без учителя. Также изучаются варианты задач и методы кластеризации с частичным обучением, в которых фиксированное подмножество объектов изначально распределено по кластерам. Задачи кластеризации вершин графа наряду с задачами о минимальном разрезе в графе являются наиболее адекватными математическими моделями задач кластеризации и классификации взаимосвязанных объектов. Однако, в отличие от задачи о минимальном разрезе в задачах кластеризации вершин графа минимизируется не только число «лишних» связей между классами, но и число «недостающих» связей внутри классов.

Актуальность темы диссертации обусловлена тем, что задачи кластеризации вершин графа являются математическими моделями множества практически важных задач социальной психологии [35], теории кодирования [41], вычислительной биологии [31, 40], кластеризации документов [23], кластеризации многомерных данных [36] и др.

Задачи кластеризации вершин графа относятся к классу NP-трудных, отыскание их точных решений представляет собой весьма сложную проблему. Поэтому возрастает актуальность построения эффективных приближенных алгоритмов и получения гарантированных оценок точности этих алгоритмов,

а также их экспериментального исследования.

Целью диссертации является исследование различных вариантов задач кластеризации вершин графа, разработка и анализ точных и приближенных алгоритмов решения этих задач.

Имеются различные подходы к анализу алгоритмов приближенного решения дискретных оптимизационных задач. В настоящей диссертации реализованы следующие два подхода: анализ алгоритма в худшем случае, когда находится гарантированная оценка точности алгоритма, справедливая для всех индивидуальных задач, и вычислительный эксперимент, использующий тестовые примеры индивидуальных задач.

В диссертации рассмотрены известные и новые варианты задач кластеризации вершин графа. Предложены новые полиномиальные алгоритмы приближенного решения различных вариантов задачи кластеризации вершин графа, получены гарантированные оценки точности этих алгоритмов. Разработаны алгоритмы нахождения точных решений этих задач. Проведено экспериментальное исследование точных и приближенных алгоритмов решения задач кластеризации вершин графа.

Приведем формулировки исследуемых задач и дадим краткий обзор известных результатов.

Постановки задач кластеризации вершин графа

Задачи кластеризации вершин графов возникают при анализе систем взаимосвязанных объектов. При этом минимизируется число связей между классами и число недостающих связей внутри классов. Постановки и различные интерпретации этих задач можно найти в работах [9, 10, 17, 23, 25, 45] и др.

Будем рассматривать только графы без петель и кратных ребер, т.е. обыкновенные графы [5]. Обыкновенный граф называется кластерным графом, если каждая его компонента связности является полным графом [40]. Обозначим через $\mathcal{M}(V)$ множество всех кластерных графов на множестве вершин V, $\mathcal{M}_k(V)$ – множество всех кластерных графов на V, имеющих ровно k компонент связности, $\mathcal{M}_{\leqslant k}(V)$ – множество всех кластерных графов на V, имеющих не более k компонент связности, $2 \leqslant k \leqslant |V|$.

Если $G_1 = (V, E_1)$ и $G_2 = (V, E_2)$ – обыкновенные помеченные графы на одном и том же множестве вершин V, то расстояние $\rho(G_1, G_2)$ между ними определяется как

$$\rho(G_1, G_2) = |E_1 \Delta E_2| = |E_1 \setminus E_2| + |E_2 \setminus E_1|,$$

т.е. $\rho(G_1, G_2)$ – это число несовпадающих ребер в графах G_1 и G_2.

Впервые задача кластеризации вершин графа была сформулированна в 1964 г. Заном под названием **задача аппроксимации графа** [45].

Задача GC (GRAPH CLUSTERING). Для произвольного графа $G = (V, E)$ найти такой граф $M^* \in \mathcal{M}(V)$, что

$$\rho(G, M^*) = \min_{M \in \mathcal{M}(V)} \rho(G, M) \stackrel{dn}{=} \tau(G).$$

В литературе рассматривались и другие варианты задачи, в которых имеются ограничения на число компонент кластерного графа.

Задача $\mathbf{GC_k}$ (k-GRAPH CLUSTERING). Дан произвольный граф $G = (V, E)$ и целое число $k, 2 \leqslant k \leqslant |V|$. Найти такой граф $M^* \in \mathcal{M}_k(V)$, что

$$\rho(G, M^*) = \min_{M \in \mathcal{M}_k(V)} \rho(G, M) \stackrel{dn}{=} \tau_k(G).$$

Задача $\mathbf{GC_{\leqslant k}}$ ($[k]$-GRAPH CLUSTERING). Дан произвольный граф $G = (V, E)$ и целое число $k, 2 \leqslant k \leqslant |V|$. Найти такой граф $M^* \in \mathcal{M}_{\leqslant k}(V)$, что

$$\rho(G, M^*) = \min_{M \in \mathcal{M}_{\leqslant k}(V)} \rho(G, M) \stackrel{dn}{=} \tau_{\leqslant k}(G).$$

Бансал, Блюм и Чаула [23] рассматривали следующую задачу, эквивалентную задаче **GC**.

Задача CC (CORRELATION CLUSTERING). Дан произвольный граф $G = (V, E)$, ребра которого имеют либо метку $+1$ (похожие), либо метку -1 (различные). Найти *оптимальную кластеризацию*, т.е. разбиение множества вершин графа G на кластеры, минимизируя *несогласованности* (количество ребер с меткой -1 внутри одного кластера плюс количество ребер с меткой $+1$ между кластерами).

В этом случае под кластеризацией понимается кластерный граф M, чьи

компоненты связности взаимно однозначно соответствуют кластерам. Вариант задачи **CC** с ограничением на число кластеров рассматриваются в [23, 40].

Рассматривались также ориентированные и взвешенные варианты задач кластеризации графов. В ориентированных вариантах каждая бикомпонента кластерного орграфа является полным симметрическим графом и отсутствуют дуги, ведущие из одной бикомпоненты в другую. Во взвешенных задачах дана весовая функция $w = V \times V \to N$ и $\rho(G_1, G_2)$ равно суммарному весу несовпадающих ребер в графах G_1 и G_2.

Оценки сложности кластеризации и критические графы

Каждая из величин $\tau(G), \tau_k(G), \tau_{\leqslant k}(G)$ называется *сложностью кластеризации графа* G в соответствующей задаче кластеризации вершин графа. Очевидно, что для любого n-вершинного графа G и $k \geqslant 2$ имеют место неравенства

$$\tau(G) \leqslant \tau_k(G) \leqslant \tau_{\leqslant k}(G) \leqslant \frac{n(n-1)}{2},$$

где последнее неравенство является тривиальной оценкой количества ребер графа.

Произвольный n-вершинный граф G называется *τ-критическим*, если он имеют наибольшую сложность кластеризации $\tau(G)$ среди всех n-вершинных графов. Понятия *τ_k-критического* и *$\tau_{\leqslant k}$-критического* графов определяется аналогично.

Следующая верхняя оценка сложности кластеризации и соответствующие ей критические графы хорошо известны. В 1974 г. Фридман [15, 17] показал, что для любого n-вершинного графа G имеет место достижимая оценка

$$\tau(G) \leqslant \left\lfloor \frac{(n-1)^2}{4} \right\rfloor.$$

С точностью до изоморфизма единственными τ-критическими n-вершинными графами является полные двудольные графы $K_n^1 = (X_1, Y_1; U_1)$ и $K_n^2 = (X_2, Y_2; U_2)$, где $0 \leqslant |X_1| - |Y_1| \leqslant 1$ и $|X_2| - |Y_2| = 2$.

Более сильный результат был установлен Томеску [42, 43]: для любого

$k \geqslant 2$ и любого n-вершинного графа G

$$\tau_{\leqslant k}(G) \leqslant \left\lfloor \frac{(n-1)^2}{4} \right\rfloor.$$

В случае $k = 2$ все полные двудольные графы являются $\tau_{\leqslant k}$-критическими (и только они), а в случае $k \geqslant 3$ графы K_n^1 и K_n^2 также являются единственными $\tau_{\leqslant k}$-критическими.

Позже Ильев и Фридман [8] установили, что для любого $k \geqslant 2$ и для любого n-вершинного графа G при $n \geqslant 5(k-1)$ справедлива достижимая оценка

$$\tau_k(G) \leqslant \left\lfloor \frac{(n-1)^2}{4} \right\rfloor.$$

Для $n \geqslant 7$ все полные двудольные графы являются τ_2-критическими (и только они), а в случае $k \geqslant 3$ и $n \geqslant 5k-1$ графы K_n^1 и K_n^2 являются единственными $\tau_{\leqslant k}$-критическими графами.

Вычислительная сложность задач кластеризации вершин графа

Первые известные результаты, касающиеся NP-трудности задачи **GC** были получены Крживанеком и Моравеком [37] в 1986 г. Через семнадцать лет Чен, Цзян и Лин [31] рассматривали задачу поиска ближайшего корня в филогенетическом дереве и также доказали NP-трудность задачи **GC**.

В последующие годы задачи кластеризации вершин графа неоднократно переоткрывались и независимо изучались под различными названиями (**задача аппроксимации графа** [1, 8, 13, 45], **Correlation Clustering** [23], **Cluster Editing** [25, 40]). В этих и других работах изучались и варианты задачи, в которых задано ограничение на количество кластеров, а также рассматривались более общие постановки.

Так, в середине 2000-х годов несколько групп исследователей независимо доказали NP-трудность различных вариантов задачи кластеризации вершин графа. В 2004 г. Бансал, Блюм и Чаула [23] свели задачу разбиения на треугольники к задаче **GC** и тем самым доказали ее NP-трудность в случае, когда все кластеры имеют мощность не меньше 3. В том же 2004 г. Ша-

мир, Шаран и Цур [40] свели задачу 3-точного покрытия к задаче **GC**. Они также свели известную NP-трудную задачу о 2-раскраске 3-равномерного гиперграфа к задаче $\mathbf{GC_2}$ и тем самым доказали, что задача $\mathbf{GC_k}$ является NP-трудной для любого фиксированного $k \geqslant 2$. В обоих случая Шамир, Шаран и Цур предложили достаточно сложное доказательство. В 2006 г. Гиотис и Гурусвами [34] опубликовали более простое доказательство этого результата путем сведения задачи о бисекции графа. В том же году Агеев, Ильев, Кононов и Талевнин [1] доказали, что задачи $\mathbf{GC_2}$ и $\mathbf{GC_{\leqslant 2}}$ NP-трудны уже на кубических графах, откуда вывели, что все упомянутые варианты задачи кластеризации вершин графа, а также их взвешенные и ориентированные аналоги являются NP-трудными.

В 2012 г. Комусевич и Улман [38] показали, что задача **GC** остается NP-трудной на графе, маскимальная степень которого равна 6. Остается открытым вопрос, является ли задача NP-трудной в случае графа, максимальная степень которого меньше 6. Бастос и др. [24] показали, что задача **GC** NP-трудна даже на графах диаметра 2.

Задача **GC** может быть решена за полиномиальное время для некоторых специальных классов графов. В 1964 г. Заном [45] была решена задача **GC** для графов, представляющих иерархические структуры. В 1971 г. Фридман [14] показал, что задача **GC** для любого графа без треугольников сводится к построению в ней наибольшего паросочетания. Этот результат доказывает, в частности, полиномиальную разрешимость задачи в случае двудольных графов. В том же году Вейнер [3] предложил алгоритм решения задачи кластеризации вершин графов, не содержащих четырехвершинных подграфов ровно с пятью ребрами. Обоснование алгоритма Вейнера было дано Фридманом [16]. Маннаа [39] предложил полиномиальный алгоритм динамического программирования для правильных интервальных графов. До сих пор остается открытым вопрос, является ли задача NP-трудной в случае общих интервальных графов. Ксин [44] разработал алгоритм с линейной трудоемкостью для задачи **GC** на графах ограниченной древесной ширины.

Приближенные алгоритмы для задач кластеризации вершин графа

Приведем необходимые понятия, характеризующие качество алгоритмов приближенного решения задач на минимум. Для задач максимизации подобные характеристики определяются аналогичным образом.

Алгоритм решения задачи комбинаторной оптимизации называется $r(n)$-*приближенным*, если он за полиномиальное время находит решение, вес которого отличается от веса оптимального решения не более, чем в $r(n)$ раз. Величина $r(n)$ называется *гарантированной оценкой точности* этого алгоритма.

Говорят, что задача комбинаторной оптимизации принадлежит классу APX, если для нее существует r-приближенный алгоритм, где r – некоторая константа.

Семейство алгоритмов H_ϵ для задачи комбинаторной оптимизации называется *полиномиальной приближенной схемой*, если для любого $\epsilon > 0$ алгоритм H_ϵ является $(1+\epsilon)$-приближенным.

Для исследования качества приближенных алгоритмов часто используется следующая идея построения алгоритмов с оценками (ϵ, δ), предложенная Гимади, Глебовым и Перепелицей [4].

Пусть $\mathcal{K}$ некоторый класс задач минимизации, $\mathcal{P}$ – семейство вероятностых мер, определенных на $\mathcal{K}$. Для $K \in \mathcal{K}$ обозначим $OPT(K)$ оптимальное значение целевой функции, а $A(K)$ – значение, найденное алгоритмом A.

Говорят, что алгоритм A имеет оценки (ϵ, δ) относительно $\mathcal{P}$, если

$$P\{A(K) > (1+\epsilon)OPT(K)\} \leqslant \delta$$

для всех $K \in \mathcal{K}$ и $P \in \mathcal{P}$. Параметры ϵ и δ могут трактоваться как оценки *относительной погрешности* и *вероятности несрабатывания* алгоритма A, соотвественно. Для подкласса $\mathcal{K}_n \subset \mathcal{K}$ задач размерности n говорят о семействах $\mathcal{P}_n$ и оценках (ϵ_n, δ_n).

Примером алгоритма с оценками (ϵ_n, δ_n) является алгоритм Боровкова [2] для класса $\mathcal{K}_n$ задач коммивояжера, в которых распределения из $\mathcal{P}_n$ получа-

ются в результате случайного выбора из n точек в ограниченной, односвязной области r-мерного евклидова пространства с достаточно гладкой границей.

Алгоритм A называется *асимптотически точным* на классе задач $\mathcal{K}$, если существуют такие последовательности (ϵ_n, δ_n), что для любого n алгоритм A имеет оценки (ϵ_n, δ_n) на подмножестве $\mathcal{K}_n \subset \mathcal{K}$, причем $\epsilon_n \to 0, \delta_n \to 0$ при $n \to \infty$.

Если при этом все $\delta_n = 0$, то такой алгоритм называется *гарантированно асимптотически точным*. Другими словами, алгоритм A *гарантированно асимптотически точен* на классе задач $\mathcal{K}$, если существует такая последовательность ϵ_n, что для любого n

$$A(K) \leqslant (1 + \epsilon_n) OPT(K)$$

на подмножестве $\mathcal{K}_n \subset \mathcal{K}$ задач размерности n, причем $\epsilon_n \to 0$ при $n \to \infty$.

Пусть $\mathcal{G}_n$ – такое семейство n-вершинных графов, что для любого графа $G = (V, E) \in \mathcal{G}_n$ выполнено $|E| \leqslant \alpha n^\beta$, где $\alpha > 0, 0 < \beta < 2$ – некоторые константы. Графы семейства $\mathcal{G}_n$ назовем *неплотными*.

Перечислим наиболее интересные результаты, касающиеся приближенных алгоритмов для задач кластеризации вершин графа.

В 2004 г. Бансал, Блюм и Чаула [23] разработали 3-приближенный алгоритм для задачи $\mathbf{GC}_{\leqslant \mathbf{2}}$.

В 2006 г. Агеев, Ильев, Кононов и Телевнин [1] доказали существование рандомизированной полиномиальной приближенной схемы для задачи $\mathbf{GC}_{\leqslant \mathbf{2}}$ путем сведения этой задачи к задаче о бисекции графа. Гиотис и Гурусвами [34] предложили рандомизированную полиномиальную приближенную схему для задачи $\mathbf{GC}_{\leqslant \mathbf{k}}$ (при любом фиксированном $k \geqslant 2$). Навроцкая, Ильев и Талевнин [11] разработали алгоритм локального поиска для задачи $\mathbf{GC}_{\leqslant \mathbf{2}}$ и показали, что если для произвольного графа $G = (V, E)$ выполняется $|E| = o(|V|^2)$, то гарантированная оценка точности алгоритма локального поиска стремится к 1 при $|V| \to \infty$. Другими словами, алгоритм локального поиска является гарантированно асимптотически точным.

В 2008 г. Коулман, Саундерсон и Вирт [32] предложили 2-приближенный алгоритм решения задачи $\mathbf{GC}_{\leqslant \mathbf{2}}$, при этом они указали на сложность поли-

номиальной приближенной схемы из [34], что лишает ее практического применения.

Для задачи $\mathbf{GC_2}$ Ильев, Ильева и Навроцкая [6] разработали $(3 - \frac{6}{|V|})$-приближенный алгоритм.

В 2005 г. Чарикар, Гурусвами и Вирт [29] доказали, что задача **GC** является APX-трудной (т.е. для нее не существует полиномиальной приближенной схемы). Они также предложили 4-приближенный алгоритм ее решения, воспользовавшись округлением LP-релаксации с методом наращивания областей. В 2008 г. Айлон, Чарикар и Ньюман [20] разработали 2,5-приближенный алгоритм для задачи **GC**. В 2015 г. Чаула, Макарычев, Схарам и Ярославцев [30] разработали для этой задачи $(2.06 - \epsilon)$-приближенный алгоритм.

Задачи и методы кластеризации частичным обучением

Методы решения задач кластеризации составляют важный раздел теории распознавания образов и машинного обучения. В машинном обучении задачи кластеризации относят к разделу *обучения без учителя*. Наряду с этим рассматриваются также *задачи и методы кластеризации с частичным обучением*, в которых часть объектов (как правило, небольшая) изначально распределена по кластерам [28, 22].

Рассмотрим следующие две формализации задачи кластеризации вершин графа с частичным обучением.

Задача $\mathbf{SGC_k}$ (k-SEMI-SUPERVISED GRAPH CLUSTERING). Дан обыкновенный граф $G=(V,E)$ и целое число k, $2 \leqslant k \leqslant |V|$. Выделено множество попарно различных вершин $Z = \{z_1, \ldots, z_k\} \subset V$. Требуется найти такой граф $M^* \in \mathcal{M}_k(V)$, что

$$\rho(G, M^*) = \min_{M \in \mathcal{M}_k(V)} \rho(G, M),$$

причем минимум берется по всем кластерным графам $M = (V, E_M) \in \mathcal{M}_k(V)$, в которых $z_i z_j \notin E_M$ для любых $i, j \in \{1, \ldots, k\}$. Другими словами, никакие две вершины множества $Z = \{z_1, \ldots, z_k\}$ не принадлежат одной и той же компоненте связности (т. е. одному кластеру) графа M.

Задача $\mathbf{SSGC_k}$ (k-SET SEMI-SUPERVISED GRAPH CLUSTERING). Дан обыкновенный граф $G = (V, E)$ и целое число k, $2 \leqslant k \leqslant |V|$. Выделено семейство $\mathcal{Z} = \{Z_1, \ldots, Z_k\}$ попарно непересекающихся непустых подмножеств множества V. Требуется найти такой граф $M^* \in \mathcal{M}_k(V)$, что

$$\rho(G, M^*) = \min_{M \in \mathcal{M}_k(V)} \rho(G, M),$$

причем минимум берется по всем кластерным графам $M = (V, E_M) \in \mathcal{M}_k(V)$, в которых

1. $zz' \notin E_M$ для всех $z \in Z_i, z' \in Z_j, i, j = 1, \ldots, k$, $i \neq j$;
2. $zz' \in E_M$ для всех $z, z' \in Z_i, i = 1, \ldots, k$.

Другими словами, все множества семейства $\mathcal{Z}$ являются подмножествами разных компонент связности (т.е. разных кластеров) графа M.

Заметим, что задача $\mathbf{SGC_k}$ является частным случаем задачи $\mathbf{SSGC_k}$ при $|Z_1| = \cdots = |Z_k| = 1$. Несложно свести по Тьюрингу задачу $\mathbf{GC_k}$ к $\mathbf{SGC_k}$ и тем самым показать, что задача $\mathbf{SGC_k}$ тоже NP-трудна. Задача $\mathbf{SSGC_k}$ является NP-трудной как обобщение задачи $\mathbf{SGC_k}$.

Прикладное значение задач кластеризации вершин графа

Одну из первых постановок задачи кластеризации вершин графа можно обнаружить в работе Харари [35] 1953 года. Им была рассмотрена следующая интерпретация: вершины графа взаимно однозначно соответствуют людям из некоторой социальной группы, а ребрами соединялись те пары, которые испытывали друг к другу симпатию. Задача состоит в том, чтобы разделить людей на две группы, минимизируя взаимную неприязь внутри каждой из групп.

Помимо социальной психологии, изучение задачи кластеризации имеет множество других приложений, в частности, в статистической механике, в которой она связана с энергетической конфигурацией модели Изинга без

внешнего поля. Сол и Заславски [41] показали связь этой задачи с теорией кодирования.

Задача кластеризации вершин графа тесно связана с биологией. Так, Чен, Цзян и Лин [31] рассматривали эту задачу как частный случай задачи поиска ближайшего корня k-ого порядка в филогенетическом дереве. Дасгупта, Энцисо, Зонтаг и Чжан [33] показали применимость этой задачи к задаче декомпозиции крупномасштабных биологических сетей на монотонные подсистемы. Шамир, Шаран и Цур [40], а также Бен-Дор, Шамир и Яхими [25] показали связь с задачами вычислительной биологии.

Разумеется, задача кластеризации вершин графа тесно связана с проблемами машинного обучения. Изучая задачи классификации документов и агностического обучения, Бансал, Блюм и Чаула [23] фактически переоткрыли задачу кластеризации вершин графа. Задача тесно связана с такими проблемами, как задача кластеризации многомерных данных[36], задачей бикластеризации [26] и др.

Структура диссертации

Диссертационная работа состоит из введения, трех глав, заключения, списка литературы и приложения.

В первой главе исследуется задача кластеризации вершин графа, в которой число кластеров не превосходит k. Приведены известные результаты для случая $k = 2$. Для задачи $\mathbf{GC_{\leqslant 3}}$ предложены два полиномиальных приближенных алгоритма. Первый алгоритм основан на известном алгоритме для задачи $\mathbf{GC_{\leqslant 2}}$, многократно применяющем процедуру локального поиска. Второй алгоритм основан на оригинальной идее и вообще не использует локальный поиск. Получены априорные гарантированные оценки точности этих алгоритмов.

Во второй главе рассматривается задача кластеризации вершин графа на два кластера и задачи и методы кластеризации с частичным обучением. Для задач $\mathbf{GC_2}$, $\mathbf{SGC_2}$ и $\mathbf{SSGC_2}$ разработана универсальная процедура локального поиска и предложено несколько приближенных алгоритмов. Доказаны априорные гарантированные оценки точности этих алгоритмов. Ис-

следованы взаимосвязи задач $\mathbf{GC_2}$, $\mathbf{SGC_2}$ и $\mathbf{SSGC_2}$.

В третьей главе предложены два подхода к нахождению оптимальных решений различных вариантов задачи кластеризации вершин графов. Первый подход использует специальную схему метода ветвей и границ, характерную для задач кластеризации вершин графа, второй подход опирается на модели целочисленного линейного программирования. Предложены эвристические алгоритмы решения задач кластеризации вершин графа. Проведено экспериментальное исследование точных и приближенных алгоритмов, целью которого было установление времени работы и точности этих алгоритмов, а также целесообразность многократного использования процедуры локального поиска.

Основные результаты работы

1. Для задачи кластеризации вершин графа, в которой число кластеров не превосходит 3, предложены два полиномиальных приближенных алгоритма. Получены априорные гарантированные оценки точности этих алгоритмов.
2. Для задачи кластеризации вершин графа на два кластера разработаны процедура локального поиска и два полиномиальных приближенных алгоритма с гарантированными оценками точности.
3. Предложены приближенные алгоритмы с частичным обучением для нового варианта задачи кластеризации вершин графа, в котором число кластеров равно 2. Получены априорные гарантированные оценки точности этих алгоритмов.
4. Предложены два точных метода решения различных вариантов задачи кластеризации вершин графа. Первый использует идею метода ветвей и границ, а второй опирается на известные и новые модели целочисленного линейного программирования рассматриваемых задач. На сериях случайных графов проведено сравнение среднего времени работы точных методов, а также экспериментальное исследование качества решений, найденных рассмотренными в работе приближенными алгоритмами.

Апробация результатов

Основные результаты диссертации докладывались на IV и V региональных конференциях магистрантов, аспирантов и молодых ученых «ФМХ ОмГУ» (Омск, 2016 и 2017); I и IV Всероссийских научно-практических конференциях «Омские научные чтения» (Омск, 2017 и 2020); VII Международной конференции «Проблемы оптимизации и их приложения (OPTA 2018)» (Омск, 2018); XVIII Международной конференции «Mathematical Optimization Theory and Operations Research (MOTOR 2019)» (Екатеринбург, 2019); XIX Международной конференции «Mathematical Optimization Theory and Operations Research (MOTOR 2020)» (Новосибирск, 2020); XX Международной конференции «Mathematical Optimization Theory and Operations Research (MOTOR 2021)» (Иркутск, 2021), на объединенном семинаре «Моделирование систем информатики» ИВМиМГ СО РАН и кафедры вычислительных систем ММФ НГУ, а также на научных семинарах в Институте математики им. С.Л. Соболева СО РАН и его Омском филиале.

Публикации

По теме диссертации автором опубликовано 11 научных работ, из них 6 статей в рецензируемых научных журналах и изданиях, определенных ВАК. В совместных работах соискателю принадлежат доказательства результатов, включенных в диссертацию. Конфликта интересов с соавторами нет.

Глава 1

Задача кластеризации вершин графа с ограниченным числом кластеров

В этой главе будет рассмотрена задача кластеризации вершин графа, в которой число кластеров не превосходит заданного числа k. Для случая $k = 2$ приведены два известных полиномиальных приближенных алгоритма с гарантированными априорными оценками точности: алгоритмы Бансала, Блюма и Чаулы [23] и Коулмана, Саундерсона и Вирта [32]. Для задачи $\mathbf{GC}_{\leqslant \mathbf{3}}$ предложены два полиномиальных приближенных алгоритма. Первый алгоритм использует в качестве процедуры алгоритм Коулмана Саундерсона и Вирта, который приближенно решает задачу $\mathbf{GC}_{\leqslant \mathbf{2}}$ и многократно применяет процедуру локального поиска. Второй алгоритм основан на оригинальной идее и вообще не использует локальный поиск. Для этих алгоритмов получены априорные гарантированные оценки точности.

1.1 Известные результаты для задачи $\mathbf{GC}_{\leqslant \mathbf{2}}$

В этом параграфе будут описаны известные результаты для задачи $\mathbf{GC}_{\leqslant \mathbf{2}}$. Приведены два полиномиальных приближенных алгоритм с гарантированными оценками точности, описана процедура локального поиска.

1.1.1 Постановка задачи и вспомогательные утверждения

Напомним постановку исследуемой задачи.

Задача $\mathbf{GC}_{\leqslant \mathbf{k}}$ ([k]-GRAPH CLUSTERING). Дан произвольный граф $G =$

(V, E) и целое число $k, 2 \leqslant k \leqslant |V|$. Найти такой граф $M^* \in \mathcal{M}_{\leqslant k}(V)$, что

$$\rho(G, M^*) = \min_{M \in \mathcal{M}_{\leqslant k}(V)} \rho(G, M).$$

Через $N_G(v)$ обозначим окрестность вершины v, т.е. множество вершин графа $G = (V, E)$, смежных с вершиной v. Через $\overline{N}_G(v)$ обозначим множество вершин графа G, не смежных с v: $\overline{N}_G(v) = V \setminus (N_G(v) \cup \{v\})$.

Пусть $G_1 = (V, E_1)$ и $G_2 = (V, E_2)$ – два графа с нумерованными вершинами на множестве V, $n = |V|$. Через $D(G_1, G_2)$ обозначим граф на множестве вершин V с множеством ребер $E_1 \Delta E_2$. Заметим, что $\rho(G_1, G_2)$ равно количеству ребер в графе $D(G_1, G_2)$.

Следующее утверждение легко доказывается с помощью леммы о рукопожатиях.

Лемма 1.1. *Пусть $d_{\min}$ – минимум степеней вершин в графе $D(G_1, G_2)$. Тогда*

$$\rho(G_1, G_2) \geqslant \frac{n d_{\min}}{2}.$$

Для множеств $V_1, \ldots V_s \subseteq V$ таких, что $V_i \cap V_j = \emptyset$ для любых $i, j \in \{1, \ldots, s\}$ и $V_1 \cup \ldots \cup V_s = V$, обозначим через $M(V_1, \ldots V_s)$ кластерный граф из множества $\mathcal{M}_{\leqslant s}(V)$ с компонентами связности, порожденными множествами $V_1, \ldots, V_s$. Сами множества $V_1, \ldots, V_s$ будем называть *кластерами*. Некоторые из V_i могут быть пустыми.

1.1.2 Приближенные алгоритмы для задачи $\mathbf{GC}_{\leqslant 2}$

Будем говорить, что кластерный граф $M(\overline{V}_1, \ldots, \overline{V}_s)$ получен из графа $M(V_1, \ldots, V_s)$ путем переноса вершины $v \in V_i$ из кластера V_i в кластер V_j, если $\overline{V}_i = V_i \setminus \{v\}$, $\overline{V}_j = V_j \cup \{v\}$ и $\overline{V}_k = V_k$ для всех $k \notin \{i, j\}$.

В 2004 г. Бансал, Блюм и Чаула [23] разработали 3-приближенный алгоритм решения задачи $\mathbf{GC}_{\leqslant 2}$.

Алгоритм BBC (Bansal-Blum-Chawla).

Вход: граф $G = (V, E)$.

Выход: кластерный граф $M_{BBC} \in \mathcal{M}_{\leqslant 2}(V)$.

Шаг 1. Для каждой вершины $v \in V$ определить кластерный граф $M_v = M(X, Y)$ следующим образом: $X = \{v\} \cup N_G(v)$, $Y = V \setminus X$ (возможно, $Y = \emptyset$).

Шаг 2. Среди всех графов M_v выбрать такой граф M_{BBC}, что

$$\rho(G, M_{BBC}) = \min_{v \in V} \rho(G, M_v).$$

Конец.

Замечание 1.1. *Трудоемкость алгоритма* **BBC** – $O(n^2)$.

Доказательство. На каждой итерации алгоритма **BBC** для произвольной вершины $v \in V$ проверяется ее смежность со всеми вершинами множества $V \setminus \{v\}$. Таким образом, трудоемкость одной итерации – $O(n)$. Количество итераций алгоритма **BBC** равно количеству вершин в графе G, т.е. не превосходит n, а значит трудоемкость алгоритма **BBC** – $O(n^2)$.

Замечание 1.1 доказано. □

Для алгоритма **BBC** справедлива следующая гарантированная оценка точности.

Теорема 1.1. *[23] Для любого графа $G = (V, E)$ имеет место неравенство*

$$\rho(G, M_{BBC}) \leqslant 3\rho(G, M^*),$$

где $M^* \in \mathcal{M}_{\leqslant 2}(V)$ – оптимальное решение задачи $\mathbf{GC}_{\leqslant \mathbf{2}}$ на графе G, а M_{BBC} – кластерный граф, построенный алгоритмом **BBC**.

Пусть $G = (V, E)$ – произвольный граф. Для вершины $v \in V$ и множества $A \subseteq V$ обозначим через A_v^+ количество таких вершин $u \in A$, что $vu \in E$. Через A_v^- обозначим число таких вершин $u \in A$, что $vu \notin E$.

В 2008 г. Коулман, Саундерсон и Вирт [32] предложили 2-приближенный алгоритм для задачи $\mathbf{GC}_{\leqslant \mathbf{2}}$, применив следующую процедуру локального поиска $\mathbf{LS}_{\leqslant \mathbf{2}}$ к каждому допустимому решению, полученному на шаге 1 алгоритма **BBC**.

Процедура $\mathbf{LS}_{\leqslant \mathbf{2}}(\mathbf{M}, \mathbf{X}, \mathbf{Y})$ (Local Search for $\mathbf{GC}_{\leqslant \mathbf{2}}$).

Вход: кластерный граф $M = M(X, Y) \in \mathcal{M}_{\leqslant 2}(V)$.
Выход: кластерный граф $\overline{M} = M(\overline{X}, \overline{Y}) \in \mathcal{M}_{\leqslant 2}(V)$.

Итерация 0. Положим $X_0 = X, Y_0 = Y$.
Итерация $\boldsymbol{k}$($\boldsymbol{k \geqslant 1}$).
Шаг 1. Для каждой вершины $u \in V$ вычислить следующую величину $\delta_k(u)$ (изменение значения целевой функции при переносе вершины u в другой кластер. При $\delta_k(u) > 0$ эту величину будем называть *локальным улучшением вершины u на итерации k*):

$$\delta_k(u) = \begin{cases} (X_{k-1})_u^- - (X_{k-1})_u^+ + (Y_{k-1})_u^+ - (Y_{k-1})_u^- & \text{для } u \in X_{k-1}, \\ (Y_{k-1})_u^- - (Y_{k-1})_u^+ + (X_{k-1})_u^+ - (X_{k-1})_u^- & \text{для } u \in Y_{k-1}. \end{cases}$$

Шаг 2. Выбрать вершину $u_k \in V$ такую, что

$$\delta_k(u_k) = \max_{u \in V} \delta_k(u).$$

Шаг 3. Если $\delta_k(u_k) \leqslant 0$, то **СТОП.** Положить $X' = X_{k-1}$, $Y' = Y_{k-1}$, $M' = M(X', Y')$. **Конец.**
Шаг 4. Если $u_k \in X_{k-1}$, то положить $X_k = X_{k-1} \setminus \{u_k\}$, $Y_k = Y_{k-1} \cup \{u_k\}$. Если же $u_k \in Y_{k-1}$, то положить $X_k = X_{k-1} \cup \{u_k\}$, $Y_k = Y_{k-1} \setminus \{u_k\}$. **Перейти на итерацию $\boldsymbol{k+1}$.**

Замечание 1.2. *Трудоемкость процедуры* $\mathbf{LS}_{\leqslant \mathbf{2}}$ – $O(n^3)$.

Доказательство. Оценим трудоемкость одной итерации процедуры $\mathbf{LS}_{\leqslant \mathbf{2}}$. На k-той итерации для каждой вершины $u \in V$ вычисляется величина $\delta_k(u)$. Заметим, что для ее вычисления достаточно знать значение величины $\delta_{k-1}(u)$ и пересчитать ее относительно вершины w, выбранной процедурой на итера-

ции $k-1$. Таким образом, трудоемкость вычисления величины $\delta_k(u)$ – $O(1)$. Следовательно, трудоемкость одной итерации процедуры $\mathbf{LS}_{\leqslant \mathbf{2}}$ – $O(n)$.

Теперь оценим количество итераций процедуры локального поиска $\mathbf{LS}_{\leqslant \mathbf{2}}$. На k-той итерации процедура либо переносит вершину $u \in V$ с максимальным значением $\delta_k(u)$ в противоположный кластер, уменьшая значение целевой функции на величину $\delta_k(u) \geqslant 1$, либо (если $\delta_k(u) \leqslant 0$ для всех вершин $u \in V$) возвращает в качестве найденного локального оптимума граф $\overline{M}(\overline{X}, \overline{Y})$, где $\overline{X} = X_{k-1}, \overline{Y} = Y_{k-1}$. Заметим, что значение целевой функции $\rho(G, M)$ удовлетворяет неравенству $0 \leqslant \rho(G, M) \leqslant \frac{n(n-1)}{2}$. Следовательно, общее количество итераций не превосходит по порядку n^2, а значит трудоемкость процедуры $\mathbf{LS}_{\leqslant \mathbf{2}}$ – $O(n^3)$.

Замечание 1.2 доказано. □

С учетом описанных выше алгоритма **BBC** и процедуры локального поиска $\mathbf{LS}_{\leqslant \mathbf{2}}$ 2-приближенный алгоритм Коулмана, Саундерсона и Вирта примет следующий вид.

Алгоритм CSW (Coleman-Saunderson-Wirth).

Вход: граф $G = (V, E)$.

Выход: кластерный граф $M_{CSW} \in \mathcal{M}_{\leqslant 2}(V)$.

Шаг 1. Пусть F – множество всех допустимых решений, построенных алгоритмом **BBC**. Применить процедуру $\mathbf{LS}_{\leqslant \mathbf{2}}$ к каждому кластерному графу из множества F.

Шаг 2. Среди всех локальных оптимумов, построенных на шаге 1, выбрать ближайший к G кластерный граф M_{CSW}.

Конец.

Замечание 1.3. *Трудоемкость алгоритма* **CSW** – $O(n^4)$.

Доказательство. На каждой итерации алгоритма **CSW** для произвольной вершины $v \in V$ проверяется ее смежность со всеми вершинами множества $V \setminus \{v\}$ за $O(n)$ операций, а также запускается процедура локального поиска $\mathbf{LS}_{\leqslant \mathbf{2}}$, трудоемкость которой – $O(n^3)$. Таким образом трудоемкость одной

итерации алгоритма **CSW** – $O(n^3)$, а значит общая трудоемкость алгоритма **CSW** – $O(n^4)$.

Замечание 1.3 доказано. □

Для алгоритма **CSW** справедлива следующая гарантированная оценка точности.

Теорема 1.2. *[32] Для любого графа $G = (V, E)$ имеет место неравенство*

$$\rho(G, M_{CSW}) \leqslant 2\rho(G, M^*),$$

где $M^* \in \mathcal{M}_{\leqslant 2}(V)$ – оптимальное решение задачи $\mathbf{GC_{\leqslant 2}}$ на графе G, а M_{CSW} – кластерный граф, построенный алгоритмом **CSW**.

1.2 Приближенные алгоритмы для задачи $\mathbf{GC_{\leqslant 3}}$

В этом параграфе будет рассмотрена задача кластеризации вершин графа не более чем на три кластера. Для этой задачи предложены два полиномиальных приближенных алгоритма. Первый, 6-приближенный алгоритм использует идеи известных алгоритмов **BBC** и **CSW**, которые приближенно решают задачу кластеризации $\mathbf{GC_{\leqslant 2}}$. Второй алгоритм основан на оригинальной идее и не использует локальный поиск. Доказаны априорные гарантированные оценки точности предложенных алгоритмов, лучшая из которых, у второго алгоритма, равна $6 - 12/n$, где n – число вершин данного графа.

1.2.1 6-приближенный алгоритм для задачи $\mathbf{GC_{\leqslant 3}}$

Используя идеи Бансала, Блюма, Чаулы [23] и Коулмана, Саундерсона, Вирта [32], можно построить следующий полиномиальный алгоритм приближенного решения задачи $\mathbf{GC_{\leqslant 3}}$.

Алгоритм $\mathbf{NLS_{\leqslant 3}}$ (Neighborhood with Local Search for $\mathbf{GC_{\leqslant 3}}$).

Вход: граф $G = (V, E)$, $|V| = n$.

Выход: кластерный граф $M_{NLS} \in \mathcal{M}_{\leqslant 3}(V)$.

Шаг 1. Если $n \leqslant 2$, то $M_{NLS} = G$, иначе переход на шаг 2.

Шаг 2. Для каждой вершины $v \in V$ выполнить:

Шаг 2.1. $V_1 = \{v\} \cup N_G(v)$. Если $V_1 = V$, то M_v – полный граф K_n, иначе переход на шаг 2.2.

Шаг 2.2. Обозначить через G_1 подграф графа G, порожденный множеством $V \setminus V_1$. Приближенно решить задачу $\mathbf{GC_{\leqslant 2}}$ на графе G_1 алгоритмом **CSW**, полученный кластерный граф обозначить через $M = M(V_2, V_3)$ (возможно, $V_3 = \emptyset$). Положить $M_v = M(V_1, V_2, V_3)$.

Шаг 3. Среди всех графов M_v выбрать ближайший к G кластерный граф M_{NLS}:

$$\rho(G, M_{NLS}) = \min_{v \in V} \rho(G, M_v).$$

Конец.

Замечание 1.4. *Трудоемкость алгоритма* $\mathbf{NLS_{\leqslant 3}}$ – $O(n^5)$.

Доказательство. На каждой итерации алгоритма $\mathbf{NLS_{\leqslant 3}}$ для произвольной вершины $v \in V$ проверяется ее смежность с вершинами множества $V \setminus \{v\}$ за $O(n)$ операций, а также в качестве процедуры вызывается алгоритм **CSW**, трудоемкость которого – $O(n^4)$. Таким образом общая трудоемкость алгоритма $\mathbf{NLS_{\leqslant 3}}$ – $O(n^5)$.

Замечание 1.4 доказано. □

Для алгоритма $\mathbf{NLS_{\leqslant 3}}$ справедлива следующая гарантированная оценка точности.

Теорема 1.3. *Для любого графа* $G = (V, E)$ *имеет место неравенство*

$$\rho(G, M_{NLS}) \leqslant 6\rho(G, M^*),$$

где $M^* \in \mathcal{M}_{\leqslant 3}(V)$ *– оптимальное решение задачи* $\mathbf{GC}_{\leqslant \mathbf{3}}$ *на графе* G*, а* M_{NLS} *– кластерный граф, построенный алгоритмом* $\mathbf{NLS}_{\leqslant \mathbf{3}}$*.*

Доказательство. Пусть v – вершина минимальной степени в графе $D = D(G, M^*)$, $d_{\min} = d_D(v)$. Рассмотрим кластерный граф $\overline{M} \in \mathcal{M}_{\leqslant 3}(V)$, полученный из M^* путем переноса $d_{\min}$ вершин в другие компоненты графа M^*: вершины множества $N_G(v) \cap N_D(v)$ переместим в компоненту, содержащую вершину v, а вершины, принадлежащие множеству $\overline{N}_G(v) \cap N_D(v)$, перенесем в любую из компонент, не содержащих v. Тогда, очевидно, кластер $V_1 = \{v\} \cup N_G(v)$ будет одним и тем же в $\overline{M}$ и в M_v.

Если $d_{\min} = 0$, то кластерный граф $\overline{M}$ совпадает с кластерным графом M^*, а значит

$$\rho(G, \overline{M}) = \rho(G, M^*) \leqslant 3\rho(G, M^*).$$

Пусть $d_{\min} > 0$. Заметим, что при переносе $d_{\min}$ вершин значение целевой функции не может увеличиться более чем на $nd_{\min}$, так как перенос одной вершины увеличивает значение целевой функции не более чем на $n = |V|$. Отсюда с учетом леммы 1.1 получаем

$$\rho(G, \overline{M}) \leqslant \rho(G, M^*) + nd_{\min} \leqslant \rho(G, M^*) + 2\rho(G, M^*) = 3\rho(G, M^*).$$

Итак,

$$\rho(G, \overline{M}) \leqslant 3\rho(G, M^*) \tag{1.1}$$

Докажем, что $\rho(G, M_v) \leqslant 6\rho(G, M^*)$. Для этого рассмотрим первый кластер $V_1 = \{v\} \cup N_G(v)$. Возможны 2 случая.

а) $V_1 = V$. В соответствии с шагом 2.1 алгоритма $\mathbf{NLS}_{\leqslant \mathbf{3}}$ полагаем $M_v = K_n = \overline{M}$, где K_n – полный n-вершинный граф. Следовательно,

$$\rho(G, M_v) = \rho(G, \overline{M}) \leqslant 3\rho(G, M^*) < 6\rho(G, M^*).$$

б) $V_1 \neq V$. Обозначим через M_1^* оптимальное решение задачи $\mathbf{GC}_{\leqslant \mathbf{2}}$ на графе G_1, порожденном множеством $V \setminus V_1$. Рассмотрим кластерный граф

$\tilde{M} = M(V_1) \cup M_1^*$, где $M(V_1)$ — полный граф на множестве V_1. Очевидно, что

$$\rho(G, \tilde{M}) \leqslant \rho(G, \overline{M}).$$

Тогда, с учетом (1.1)

$$\rho(G, \tilde{M}) \leqslant 3\rho(G, M^*). \tag{1.2}$$

Пусть $M = M(V_2, V_3)$ – допустимое решение задачи $\mathbf{GC}_{\leqslant \mathbf{2}}$ на графе G_1, найденное алгоритмом **CSW** на шаге 2.2, если на шаге 2.1 выбрана вершина v. Тогда $M_v = M(V_1, V_2, V_3)$ (множество V_3 может быть пустым).

Обозначим через s сумму числа отсутствующих ребер в подграфе графа G, порожденном множеством вершин $V_1 = \{v\} \cup N_G(v)$, и величины разреза $(V_1, V \setminus V_1)$ в графе G.

Очевидно, что

$$\rho(G, M_v) = s + \rho(G_1, M),\ \rho(G, \tilde{M}) = s + \rho(G_1, M_1^*). \tag{1.3}$$

Если $|V \setminus V_1| \leqslant 2$, то $M = G_1$ и $\rho(G_1, M) = \rho(G_1, M_1^*) = 0$. Тогда, с учетом (1.2) и (1.3)

$$\rho(G, M_v) = \rho(G, \tilde{M}) \leqslant 3\rho(G, M^*) < 6\rho(G, M^*).$$

Пусть $|V \setminus V_1| \geqslant 3$. Тогда по теореме 1.2

$$\rho(G_1, M) \leqslant 2\rho(G_1, M_1^*).$$

В силу (1.2) и (1.3) получим

$$\begin{aligned} \rho(G, M_v) = s + \rho(G_1, M) &\leqslant s + 2\rho(G_1, M_1^*) \leqslant 2s + 2\rho(G_1, M_1^*) \\ &= 2(s + \rho(G_1, M_1^*)) = 2\rho(G, \tilde{M}) \leqslant 6\rho(G, M^*). \end{aligned}$$

На шаге 2 алгоритма $\mathbf{NLS}_{\leqslant \mathbf{3}}$ среди всех графов будет рассмотрен и граф M_v, где v – вершина минимальной степени в графе $D = D(G, M^*)$. Отсюда и получается требуемая оценка точности алгоритма $\mathbf{NLS}_{\leqslant \mathbf{3}}$.

Теорема 1.3 доказана. □

1.2.2 Приближенный алгоритм для задачи $\mathbf{GC}_{\leqslant 3}$, не использующий локальный поиск

Представим еще один полиномиальный алгоритм приближенного решения задачи $\mathbf{GC}_{\leqslant 3}$ с лучшей гарантированной оценкой точности, основанный на другой идее и не использующий локальный поиск. Нам потребуется следующее вспомогательное утверждение.

Лемма 1.2. *Пусть $G = (V, E)$ – n-вершинный граф, $M = M(V_1, \ldots, V_s) \in \mathcal{M}_{\leqslant s}(V)$, $n \geqslant 3$, а вершина $v \in V_i$ такая, что ее окрестность в графе $D = D(G, M)$ не пуста. Рассмотрим кластерный граф $M' \in \mathcal{M}_{\leqslant s}(V)$, полученный из графа M следующим образом: либо произвольная вершина $u \in N_G(v) \cap N_D(v)$ перенесена в кластер V_i, либо произвольная вершина $u \in \overline{N}_G(v) \cap N_D(v)$ перенесена в любой из кластеров V_j, не содержащий вершину v. Тогда*

$$\rho(G, M') \leqslant \rho(G, M) + (n - 3).$$

Доказательство. Итак, пусть $u \in N_D(v)$. Очевидно, что графы D и $D' = D(G, M')$ будут отличаться только ребрами вида $uw, w \in V$, остальные ребра у них совпадают. Следовательно,

$$\rho(G, M') - \rho(G, M) = |N_{D'}(u)| - |N_D(u)|.$$

Так как $v \in N_D(u)$, а $N_{D'}(u) \subseteq V \setminus \{u, v\}$, то $|N_D(u)| \geqslant 1$, $|N_{D'}(u))| \leqslant n - 2$, откуда

$$\rho(G, M') - \rho(G, M) \leqslant n - 2 - 1 = n - 3.$$

Лемма 1.2 доказана. □

Пусть $M^* = M(V_1^*, V_2^*, V_3^*) \in \mathcal{M}_{\leqslant 3}(V)$ – оптимальное решение задачи $\mathbf{GC}_{\leqslant 3}$ на графе $G = (V, E)$, причем $M^* \neq K_n$, где $n = |V|$. Через $d_D(v)$ обозначим степень вершины v в графе $D = D(G, M^*)$.

Положим $d_1 = \min\{d_D(v) : v \in V\} = d_D(v_1)$, и через V_1^* обозначим тот кластер графа M^*, который содержит вершину v_1, $n_1 = |V_1^*|$. Пусть $d_2 = \min\{d_D(v) : v \in V \setminus V_1^*\} = d_D(v_2)$, и через V_2^* обозначим тот кластер графа M^*, который содержит вершину v_2. Очевидно, $d_1 \leqslant d_2$.

Так как $\rho(G, M^*)$ равно числу ребер в графе $D(G, M^*)$, то из леммы о рукопожатиях и леммы 1 вытекает следующее утверждение.

Лемма 1.3. *Для произвольного n-вершинного графа $G = (V, E)$ имеет место неравенство*

$$\rho(G, M^*) \geqslant \frac{d_1 n_1 + d_2(n - n_1)}{2} \geqslant \frac{n d_1}{2},$$

где M^* – оптимальное решение задачи $\mathbf{GC}_{\leqslant \mathbf{3}}$ на графе G.

Следующий алгоритм основан на идее, отличной от той, что была использована в [23, 32] для приближенного решения задачи $\mathbf{GC}_{\leqslant \mathbf{2}}$.

Алгоритм $\mathbf{DN}_{\leqslant \mathbf{3}}$ (Double Neighbourhood for $\mathbf{GC}_{\leqslant \mathbf{3}}$).

Вход: граф $G = (V, E)$, $n = |V|$, $n \geqslant 3$.
Выход: кластерный граф $M_{DN} \in \mathcal{M}_{\leqslant 3}(V)$.

Шаг 1. Для каждой упорядоченной пары вершин $(v, w) \in V \times V$ такой, что $v \neq w$, выполнить:
Шаг 1.1. Положить $V_1 = \{v\} \cup (N_G(v) \setminus \{w\})$. Переход на шаг 1.2.
Шаг 1.2. Обозначить через G_1 подграф графа G, порожденный множеством $V \setminus V_1$. Положить $V_2 = \{w\} \cup N_{G_1}(w)$, $V_3 = V \setminus (V_1 \cup V_2)$ (возможно, $V_3 = \emptyset$). Положить $M_{vw} = M(V_1, V_2, V_3)$.
Шаг 2. Среди построенных графов M_{vw} и графа K_n выбрать ближайший к G кластерный граф $M_{DN} \in \mathcal{M}_{\leqslant 3}(V)$:

$$\rho(G, M_{DN}) = \min_{\substack{(v, w) \in V \times V, \\ v \neq w}} \{\rho(G, M_{vw}), \rho(G, K_n)\}.$$

Конец.

Замечание 1.5. *Трудоемкость алгоритма $\mathbf{DN}_{\leqslant \mathbf{3}}$ – $O(n^3)$.*

Доказательство. На каждой итерации алгоритма $\mathbf{DN}_{\leqslant \mathbf{3}}$ для произвольной пары вершин $(v, w) \in V \times V$ проверяется смежность этих вершин с другими вершинами множества $V \setminus \{v, w\}$ за $O(n)$ операций. Поскольку количество

итераций алгоритма равно количеству упорядоченных пар вершин в графе $G = (V, E)$, то общая трудоемкость алгоритма $\mathbf{DN}_{\leqslant \mathbf{3}}$ – $O(n^3)$.

Замечание 1.5 доказано. □

Для алгоритма $\mathbf{DN}_{\leqslant \mathbf{3}}$ справедлива следующая гарантированная оценка точности.

Теорема 1.4. *При $n \geqslant 3$ для любого n-вершинного графа $G = (V, E)$ имеет место неравенство*

$$\rho(G, M_{DN}) \leqslant (6 - \frac{12}{n})\rho(G, M^*),$$

где $M^* \in \mathcal{M}_{\leqslant 3}(V)$ – оптимальное решение задачи $\mathbf{GC}_{\leqslant \mathbf{3}}$ на графе G, а M_{DN} – кластерный граф, построенный алгоритмом $\mathbf{DN}_{\leqslant \mathbf{3}}$.

Доказательство. Возможны 2 случая.

1) $M^* = K_n$. Тогда на шаге 2 алгоритма $\mathbf{DN}_{\leqslant \mathbf{3}}$ в качестве M_{DN} будет выбран граф K_n, следовательно,

$$\rho(G, M_{DN}) = \rho(G, M^*) \leqslant (6 - \frac{12}{n})\rho(G, M^*).$$

2) $M^* = M(V_1^*, V_2^*, V_3^*) \neq K_n$. Пусть как и ранее $v_1, v_2 \in V$ такие вершины, что $d_D(v_1) = d_1$, $d_D(v_2) = d_2$, причем $v_1 \in V_1^*, v_2 \in V_2^*$. Рассмотрим кластерный граф $M \in \mathcal{M}_{\leqslant 3}(V)$, полученный из графа M^* путем переноса не более чем d_1 вершин в другие кластеры графа M^*: вершины множества $(N_G(v_1) \setminus \{v_2\}) \cap N_D(v_1)$ перенесены в кластер, содержащий вершину v_1 (т.е. в кластер V_1^*), а вершины множества $\overline{N}_G(v_1) \cap N_D(v_1)$ перенесены в кластер, не содержащий ни v_1, ни v_2 (т.е. в кластер V_3^*). Пусть $M = M(V_1, V_2, V_3)$, где $V_1 = \{v_1\} \cup (N_G(v_1) \setminus \{v_2\})$ и V_2 – кластер, содержащий вершину v_2.

Если $d_1 > 0$ то, применив не более чем d_1 раз лемму 1.2, получим

$$\rho(G, M) \leqslant \rho(G, M^*) + d_1(n - 3).$$

Если же $d_1 = 0$, то граф M совпадает с графом M^*, а значит,

$$\rho(G, M) = \rho(G, M^*) = \rho(G, M^*) + d_1(n - 3).$$

Итак, показано, что в любом случае

$$\rho(G, M) \leqslant \rho(G, M^*) + d_1(n-3). \tag{1.4}$$

Заметим, что $V_1 \neq V$, поскольку $v_2 \notin V_1$. Пусть G_1 – подграф графа G, порожденный множеством вершин $V \setminus V_1$. Рассмотрим кластерный граф $M_{v_2} \in \mathcal{M}_{\leqslant 2}(V \setminus V_1)$, построенный следующим образом: вершина v_2 и все смежные с ней вершины графа G_1 принадлежат одному кластеру графа M_{v_2}, а все несмежные с v_2 вершины – другому кластеру графа M_{v_2} (этот кластер может быть пустым).

Пусть $M_1 \in \mathcal{M}_{\leqslant 2}(V \setminus V_1)$ – подграф графа M, порожденный множеством вершин $V \setminus V_1$, т.е. $M_1 = M(V_2, V_3)$. Положим $D_1 = D(G_1, M_1)$. Обозначим через $\overline{d}_2$ степень вершины v_2 в графе D_1. Очевидно, что кластерный граф M_{v_2} получен из графа M_1 путем переноса $\overline{d}_2$ вершин: вершины множества $N_{G_1}(v_2) \cap N_{D_1}(v_2)$ перенесены в кластер, содержащий вершину v_2 (т.е. в компоненту, порожденную множеством V_2), а все вершины множества $\overline{N}_{G_1}(v_2) \cap N_{D_1}(v_2)$ перенесены в кластер, не содержащий v_2 (т.е. в компоненту, порожденную множеством V_3).

Если $|V \setminus V_1| \leqslant 2$, то, очевидно, $M_{v_2} = G_1$ и $\rho(G_1, M_{v_2}) = 0$. Если при этом $\overline{d}_2 = 0$ то $M_{v_2} = M_1$. Таким образом,

$$\rho(G_1, M_{v_2}) = \rho(G_1, M_1) = \rho(G_1, M_1) + \overline{d}_2(|V \setminus V_1| - 3).$$

Пусть теперь $\overline{d}_2 > 0$. Это возможно только при $|V \setminus V_1| = 2$. В этом случае $\overline{d}_2 = 1$. Это означает, что $\rho(G_1, M_1) = 1$. Тогда

$$\rho(G_1, M_{v_2}) = 0 = \rho(G_1, M_1) - 1 \leqslant \rho(G_1, M_1) + \overline{d}_2(|V \setminus V_1| - 3).$$

Теперь рассмотрим случай $|V \setminus V_1| \geqslant 3$. Если $\overline{d}_2 > 0$ то, применив $\overline{d}_2$ раз лемму 1.2, получим

$$\rho(G_1, M_{v_2}) \leqslant \rho(G_1, M_1) + \overline{d}_2(|V \setminus V_1| - 3).$$

Если же $\overline{d}_2 = 0$, то графы M_{v_2} и M_1 совпадают, следовательно,

$$\rho(G_1, M_{v_2}) = \rho(G_1, M_1) = \rho(G_1, M_1) + \overline{d}_2(|V \setminus V_1| - 3).$$

Итак, показано, что в любом случае

$$\rho(G_1, M_{v_2}) \leqslant \rho(G_1, M_1) + \overline{d}_2(|V \setminus V_1| - 3).$$

Так как граф M был получен из графа M^* путем переноса не более чем d_1 вершин в другие кластеры, то $|V_1| \leqslant |V_1^*| - d_1 = n_1 - d_1$, откуда

$$\rho(G_1, M_{v_2}) \leqslant \rho(G_1, M_1) + \overline{d}_2(n - n_1 + d_1 - 3). \tag{1.5}$$

Покажем, что $\overline{d}_2 \leqslant d_2$. В самом деле, $d_2 = |N_D(v_2)|$, $\overline{d}_2 = |N_{D_1}(v_2)|$. Множество $N_D(v_2)$ состоит из всех вершин u таких, что либо $u \in N_G(v_2) \cap (V_1^* \cup V_3^*)$, либо $u \in \overline{N}_G(v_2) \cap V_2^*$, а множество $N_{D_1}(v_2)$ – из всех вершин u таких, что либо $u \in N_{G_1}(v_2) \cap V_3$, либо $u \in \overline{N}_{G_1}(v_2) \cap V_2$, т.е.

$$N_D(v_2) = (N_G(v_2) \cap (V_1^* \cup V_3^*)) \cup (\overline{N}_G(v_2) \cap V_2^*),$$

$$N_{D_1}(v_2) = (N_{G_1}(v_2) \cap V_3) \cup (\overline{N}_{G_1}(v_2) \cap V_2).$$

Очевидно, что $V_3 \subseteq V_1^* \cup V_3^*$ и $V_2 \subseteq V_2^*$, а так как G_1 – подграф графа G, то $N_{G_1}(v_2) \subseteq N_G(v_2)$ и $\overline{N}_{G_1}(v_2) \subseteq \overline{N}_G(v_2)$. Следовательно,

$$N_{G_1}(v_2) \cap V_3 \subseteq N_G(v_2) \cap (V_1^* \cup V_3^*),$$

$$\overline{N}_{G_1}(v_2) \cap V_2 \subseteq \overline{N}_G(v_2) \cap V_2^*.$$

Значит, $N_{D_1}(v_2) \subseteq N_D(v_2)$, откуда получаем $\overline{d}_2 = |N_{D_1}(v_2)| \leqslant |N_D(v_2)| = d_2$.

Итак, $\overline{d}_2 \leqslant d_2$, и тогда из (1.5) следует, что

$$\rho(G_1, M_{v_2}) \leqslant \rho(G_1, M_1) + d_2(n - n_1 + d_1 - 3). \tag{1.6}$$

В соответствии с шагом 2 алгоритма $\mathbf{DN}_{\leqslant \mathbf{3}}$ положим $M_{v_1 v_2} = M(V_1) \cup M_{v_2}$, где $M(V_1)$ – полный граф на множестве V_1. Обозначим через s сумму числа отсутствующих ребер в подграфе графа G, порожденном множеством вершин

V_1, и величины разреза $(V_1, V \setminus V_1)$ в графе G. Очевидно, что

$$\rho(G, M_{v_1v_2}) = s + \rho(G_1, M_{v_2}),\ \rho(G, M) = s + \rho(G_1, M_1).$$

Отсюда с учетом (1.4) и (1.6) имеем

$$\rho(G, M_{v_1v_2}) = s + \rho(G_1, M_{v_2}) \leqslant s + \rho(G_1, M_1) + d_2(n - n_1 + d_1 - 3) =$$

$$\rho(G, M) + d_2(n - n_1 + d_1 - 3) \leqslant \rho(G, M^*) + d_1(n - 3) + d_2(n - n_1 + d_1 - 3),$$

т.е.

$$\rho(G, M_{v_1v_2}) \leqslant \rho(G, M^*) + d_1(n - 3) + d_2(n - n_1 + d_1 - 3). \qquad (1.7)$$

Теперь покажем, что $d_2 \leqslant n_1 + \frac{n}{2}$. Действительно,

$$d_2 = |N_D(v_2| = |N_D(v_2)) \cap V_1^*| + |N_D(v_2) \cap (V_2^* \cup V_3^*|.$$

Очевидно, что $|N_D(v_2) \cap V_1^*| \leqslant |V_1^*| = n_1$. Обозначим $d' = |N_D(v_2) \cap (V_2^* \cup V_3^*)|$ и докажем, что $d' \leqslant \frac{n}{2}$. Предположим противное, т.е. $d' > \frac{n}{2}$. Рассмотрим кластерный граф $\tilde{M}$, полученный из M^* путем переноса вершины v_2 из кластера V_2^* в кластер V_3^*. Очевидно, что $V_2 \cup V_3 = V_2^* \cup V_3^*$ и графы D и $\tilde{D} = D(G, \tilde{M})$ отличаются только ребрами вида (u, v_2), где $u \in V_2^* \cup V_3^*$, остальные ребра у них совпадают. Следовательно,

$$\rho(G, \tilde{M}) - \rho(G, M^*) = |N_{\tilde{D}}(v_2) \cap (V_2^* \cup V_3^*)| - |N_D(v_2) \cap (V_2^* \cup V_3^*)|.$$

Так как

$$N_{\tilde{D}}(v_2) \cap (V_2^* \cup V_3^*) = (V_2^* \cup V_3^*) \setminus (\{v_2\} \cup N_D(v_2))$$
$$= (V_2^* \cup V_3^*) \setminus ((\{v_2\} \cup N_D(v_2)) \cap (V_2^* \cup V_3^*)),$$

то

$$|N_{\tilde{D}}(v_2) \cap (V_2^* \cup V_3^*)| - |N_D(v_2) \cap (V_2^* \cup V_3^*)| = (n - n_1 - d' - 1) - d' = n - n_1 - 2d' - 1.$$

По предположению, $d' > \frac{n}{2}$, поэтому $\rho(G, \tilde{M}) - \rho(G, M^*) < -n_1 - 1 < 0$, что противоречит оптимальности кластерного графа M^*. Следовательно, $d' \leqslant \frac{n}{2}$

откуда

$$d_2 \leqslant n_1 + \frac{n}{2}. \tag{1.8}$$

Оценим слагаемые из правой части неравенства (1.7) с учетом (1.8), леммы 1.3 и неравенства $d_1 \leqslant d_2$.

а) Дадим оценку суммы $\rho(G, M^*) + d_1(n-3)$:

$$\rho(G, M^*) + d_1(n-3) = \rho(G, M^*) + d_1 n(1 - \frac{3}{n})$$
$$\leqslant \rho(G, M^*) + 2\rho(G, M^*)(1 - \frac{3}{n}) = (3 - \frac{6}{n})\rho(G, M^*).$$

б) Оценим слагаемое $d_2(n - n_1 + d_1 - 3)$:

$$d_2(n - n_1 + d_1 - 3) = d_1 n_1 - d_1 n_1 + d_2(n - n_1) + d_1 d_2 - 3d_2$$
$$\leqslant 2\rho(G, M^*) - d_1 n_1 + d_1 d_2 - 3d_2 \leqslant 2\rho(G, M^*) - d_1 n_1 + d_1(n_1 + \frac{n}{2}) - 3d_2$$
$$= 2\rho(G, M^*) + \frac{d_1 n}{2} - 3d_2 \leqslant 2\rho(G, M^*) + \frac{d_1 n}{2} - 3d_1$$
$$= 2\rho(G, M^*) + \frac{d_1 n}{2}(1 - \frac{6}{n}) \leqslant 2\rho(G, M^*) + \rho(G, M^*)(1 - \frac{6}{n}) = (3 - \frac{6}{n})\rho(G, M^*).$$

Итак,

$$\rho(G, M_{v_1 v_2}) \leqslant (6 - \frac{12}{n})\rho(G, M^*).$$

На шаге 2 алгоритмом $\mathbf{DN}_{\leqslant \mathbf{3}}$ среди всех графов будет рассмотрен и граф $M_{v_1 v_2}$. Отсюда и получается требуемая оценка точности алгоритма $\mathbf{DN}_{\leqslant \mathbf{3}}$.

Теорема 1.4 доказана. □

Глава 2

Задачи кластеризации вершин графа с фиксированным числом кластеров

К задачам кластеризации вершин графа с фиксированным числом кластеров относятся:

- задача кластеризации вершин графа, в которой число кластеров равно k,
- задачи кластеризации вершин графа с частичным обучением.

В этой главе исследуются три такие задачи: $\mathbf{GC_2}$, $\mathbf{SGC_2}$ и $\mathbf{SSGC_2}$. Для каждой из этих задач предложена по два полиномиальных приближенных алгоритма и доказаны их априорные гарантированные оценки точности. Разработана универсальная процедура локального поиска, подходящая для любой из этих задач. Исследованы взаимосвязи между этими задачами.

2.1 Задача $\mathbf{GC_2}$

В этом параграфе будут приведены основные результаты о задаче кластеризации вершин графа на два кластера. Описана универсальная процедура локального поиска, подходящая для любой из задач кластеризации вершин графа с числом кластеров равным 2. Предложены два полиномиальных приближенных алгоритма решения этой задачи. Доказаны априорные гарантированные оценки точности этих алгоритмов.

2.1.1 Постановка задачи и вспомогательные утверждения

Рассмотрим частный случай задачи $\mathbf{GC_k}$ при $k = 2$.

Задача $\mathbf{GC_2}$ (2-GRAPH-CLUSTERING). Для произвольного графа $G = (V, E)$ найти такой граф $M^* \in \mathcal{M}_2(V)$, что

$$\rho(G, M^*) = \min_{M \in \mathcal{M}_2(V)} \rho(G, M).$$

Напомним, что для произвольного графа $G = (V, E)$, вершины $v \in V$ и множества $A \subseteq V$ через A_v^+ обозначается количество таких вершин $u \in A$, что $vu \in E$. Через A_v^- обозначается число таких вершин $u \in A$, что $vu \notin E$.

Докажем следующую лемму, справедливую для любого графа $G = (V, E)$ и всех кластерных графов из множества $\mathcal{M}_2(V)$.

Лемма 2.1. *Пусть $G = (V, E)$ – произвольный граф, а $M_1 = M(X_1, Y_1)$ и $M_2 = M(X_2, Y_2)$ – два произвольных кластерных графа из множества $\mathcal{M}_2(V)$. Тогда*

$$\begin{aligned} &\rho(G, M_1) - \rho(G, M_2) \\ &= \sum_{u \in X_1 \cap Y_2} \Big((X_1 \cap X_2)_u^- - (X_1 \cap X_2)_u^+ + (Y_1 \cap Y_2)_u^+ - (Y_1 \cap Y_2)_u^- \Big) \\ &\quad + \sum_{u \in Y_1 \cap X_2} \Big((Y_1 \cap Y_2)_u^- - (Y_1 \cap Y_2)_u^+ + (X_1 \cap X_2)_u^+ - (X_1 \cap X_2)_u^- \Big). \end{aligned} \tag{2.1}$$

Доказательство. Заметим, что $X_2 = (X_1 \cap X_2) \cup (Y_1 \cap X_2), Y_2 = (X_1 \cap Y_2) \cup (Y_1 \cap Y_2)$.

Тогда расстояние между G и M_2 равно

$$\begin{aligned} \rho(G, M_2) = &\frac{1}{2} \sum_{u \in X_1 \cap X_2} (X_1 \cap X_2)_u^- + \frac{1}{2} \sum_{u \in Y_1 \cap X_2} (Y_1 \cap X_2)_u^- \\ &+ \frac{1}{2} \sum_{u \in X_1 \cap Y_2} (X_1 \cap Y_2)_u^- \frac{1}{2} \sum_{u \in Y_1 \cap Y_2} (Y_1 \cap Y_2)_u^- \\ &+ \sum_{u \in X_1 \cap Y_2} (X_1 \cap X_2)_u^+ + \sum_{u \in Y_1 \cap X_2} (Y_1 \cap Y_2)_u^+ + \sum_{u \in Y_1 \cap X_2} (X_1 \cap X_2)_u^- \\ &+ \sum_{u \in Y_1 \cap X_2} (X_1 \cap Y_2)_u^+ + \sum_{u \in Y_1 \cap Y_2} (X_1 \cap X_2)_u^+ + \sum_{u \in X_1 \cap Y_2} (Y_1 \cap Y_2)_u^-. \end{aligned} \tag{2.2}$$

Аналогично, расстояние между G и M_1 равно

$$\begin{aligned}\rho(G,M_1) = \frac{1}{2}\sum_{u\in X_1\cap X_2}(X_1\cap X_2)_u^- + \frac{1}{2}\sum_{u\in Y_1\cap X_2}(Y_1\cap X_2)_u^- \\ +\frac{1}{2}\sum_{u\in X_1\cap Y_2}(X_1\cap Y_2)_u^- + \frac{1}{2}\sum_{u\in Y_1\cap Y_2}(Y_1\cap Y_2)_u^- \\ +\sum_{u\in X_1\cap Y_2}(X_1\cap X_2)_u^- + \sum_{u\in Y_1\cap X_2}(Y_1\cap Y_2)_u^- + \sum_{u\in Y_1\cap X_2}(X_1\cap X_2)_u^+ \\ +\sum_{u\in Y_1\cap X_2}(X_1\cap Y_2)_u^+ + \sum_{u\in Y_1\cap Y_2}(X_1\cap X_2)_u^+ + \sum_{u\in X_1\cap Y_2}(Y_1\cap Y_2)_u^+.\end{aligned} \tag{2.3}$$

Вычитая (2.2) из (2.3), получим требуемое равенство.

Лемма 2.1 доказана. □

2.1.2 3-приближенный алгоритм для задачи $\mathbf{GC_2}$

Построим следующий приближенный алгоритм для задачи $\mathbf{GC_2}$. В отличие от 3-приближенного алгоритма **BBC** для задачи $\mathbf{GC_{\leqslant 2}}$, данный алгоритм рассматривает лишь кластерные графы из множества $\mathcal{M}_2$ (т.е. только допустимые решения задачи $\mathbf{GC_2}$).

Алгоритм $\mathbf{N_2}$ (Neighbourhood for $\mathbf{GC_2}$).

Вход: граф $G = (V, E)$.
Выход: кластерный граф $M_N = M(X, Y) \in \mathcal{M}_2(V)$.

Шаг 1. Для каждой упорядоченной пары вершин $(v, w) \in V \times V, v \neq w$, построить кластерный граф $M_{vw} = M(X, Y)$, где $X = \{v\} \cup (N_G(v) \setminus \{w\}), Y = V \setminus X$.

Шаг 2. Среди всех кластерных графов M_{vw} выбрать ближайший к G кластерный граф M_N:

$$\rho(G, M_N) = \min_{\substack{(v,w)\in V\times V,\\ v\neq w}} \rho(G, M_{vw}).$$

Конец.

Алгоритм $\mathbf{N_2}$ отличается от алгоритма **BBC** тем, что исключает вершину $w \neq v$ из множества X.

Замечание 2.1. *Трудоемкость алгоритма* $\mathbf{N_2}$ – $O(n^3)$.

Доказательство трудоемкости алгоритма $\mathbf{N_2}$ аналогично доказательству трудоемкости алгоритма $\mathbf{DN_{\leqslant 3}}$ из раздела 1.2.2.

Для алгоритма $\mathbf{N_2}$ справедлива следующая гарантированная оценка точности.

Теорема 2.1. *Для любого графа* $G = (V, E)$ *имеет место неравенство*

$$\rho(G, M_N) \leqslant 3\rho(G, M^*),$$

где $M^* \in \mathcal{M}_2(V)$ *– оптимальное решение задачи* $\mathbf{GC_2}$ *на графе* G, *а* M_N *– кластерный граф, построенный алгоритмом* $\mathbf{N_2}$.

Доказательство. Пусть v – вершина минимальной степени в графе $D = D(G, M^*)$, т.е. $d_D(v) = d_{\min}$. Без ограничения общности будем считать, что $v \in X^*$. Тогда, очевидно, существует такая вершина w, что $w \in Y^*$. Рассмотрим кластерный граф $M_{vw} = M(X, Y)$, построенный алгоритмом $\mathbf{N_2}$ на шаге 1. Докажем, что этот граф может быть получен из графа M^* путем переноса не более чем $d_{\min}$ вершин в другой кластер.

По определению графа D

$$X^* = \{v\} \cup (N_G(v) \setminus N_D(v)) \cup (\overline{N}_G(v) \cap N_D(v)). \tag{2.4}$$

Нетрудно заметить, что

$$N_G(v) = (N_G(v) \setminus N_D(v)) \cup (N_G(v) \cap N_D(v)).$$

Возможны два случая.

Случай 1. Вершины v и w не смежны в G, т.е., $w \in \overline{N}_G(v) \cap \overline{N}_D(v)$. Тогда $N_G(v) \setminus \{w\} = N_G(v)$. Вычислим мощность множества $X^* \Delta X$. По определе-

нию графа M_{vw},

$$X = \{v\} \cup (N_G(v) \setminus \{w\}) = \{v\} \cup N_G(v)$$
$$= \{v\} \cup (N_G(v) \setminus N_D(v)) \cup (N_G(v) \cap N_D(v)).$$

Тогда, используя (2.4), мы получим, что

$$X^* \Delta X = (X^* \setminus X) \cup (X \setminus X^*) = (\overline{N}_G(v) \cap N_D(v)) \cup (N_G(v) \cap N_D(v)) = N_D(v).$$

Таким образом, $|X^* \Delta X| = |N_D(v)| = d_D(v) = d_{\min}$, а значит граф M_{vw} может быть получен из графа M^* путем переноса $d_{\min}$ вершин множества $N_D(v)$ в другой кластер.

Случай 2. Вершины v и w смежны в G, т.е., $w \in N_G(v) \cap N_D(v)$. Тогда $d_{\min} \geqslant 1$. Вычислим мощность множества $X^* \Delta X$. По определению графа M_{vw},

$$X = \{v\} \cup (N_G(v) \setminus \{w\}) = (\{v\} \cup N_G(v)) \setminus \{w\}$$
$$= (\{v\} \cup (N_G(v) \setminus N_D(v)) \cup (N_G(v) \cap N_D(v))) \setminus \{w\}.$$

Тогда, используя (2.4) и включение $w \in N_G(v) \cap N_D(v)$, мы получим, что

$$X^* \Delta X = \Big((\overline{N}_G(v) \cap N_D(v)) \cup (N_G(v) \cap N_D(v)) \Big) \setminus \{w\} = N_D(v) \setminus \{w\}.$$

Таким образом, $|X^* \Delta X| = |N_D(v)| - 1 = d_{\min} - 1$, а значит, граф M_{vw} может быть получен из графа M^* путем переноса $d_{\min} - 1$ вершин множества $N_D(v) \setminus \{w\}$ в другой кластер.

Итак, мы показали, что граф M_{vw} может быть получен из графа M^* путем переноса не более чем $d_{\min}$ вершин в другой кластер. При переносе $d_{\min}$ вершин значение целевой функции не может увеличиться более чем на $nd_{\min}$, так как перенос одной вершины увеличивает значение целевой функции не более чем на $n = |V|$. Отсюда с учетом леммы 1.1 получаем

$$\rho(G, M_{vw}) \leqslant \rho(G, M^*) + nd_{\min} \leqslant \rho(G, M^*) + 2\rho(G, M^*) = 3\rho(G, M^*).$$

На шаге 1 алгоритмом $\mathbf{N_2}$ среди всех графов будет рассмотрен и граф M_{vw}. Отсюда и получается требуемая оценка точности алгоритма $\mathbf{N_2}$.

Теорема 2.1 доказана. □

Следствие 2.1. *Среди всех графов, построенных алгоритмом* $\mathbf{N_2}$ *на шаге 1, всегда существует такой граф* $M_{vw} = M(X, Y)$, *что*

1. M_{vw} *может быть получен из графа* M^* *путем переноса не более чем* $d_{\min} = d_D(v)$ *вершин (здесь* $D = D(G, M^*)$*),*
2. $v \in X \cap X^*, w \in Y \cap Y^*$.

2.1.3 Процедура локального поиска

Рассмотрим следующую процедуру локального поиска.

Процедура $\mathbf{LS_2}(\mathbf{M}, \mathbf{X}, \mathbf{Y}, \mathbf{Z_1}, \mathbf{Z_2})$ (Local search for 2 components).

Вход: кластерный граф $M = M(X, Y) \in \mathcal{M}_2(V)$, $Z_1 \subset X, Z_2 \subset Y$.
Выход: кластерный граф $M' = M(X', Y') \in \mathcal{M}_2(V)$.

Итерация 0. Положим $X_0 = X, Y_0 = Y$.
Итерация k($k \geqslant 1$).
Шаг 1. Для каждой вершины $u \in V \setminus (Z_1 \cup Z_2)$ вычислить следующую величину $\delta_k(u)$ (изменение значения целевой функции при переносе вершины u в другой кластер. При $\delta_k(u) > 0$ эту величину будем называть *локальным улучшением вершины* u *на итерации* k):

$$\delta_k(u) = \begin{cases} (X_{k-1})_u^- - (X_{k-1})_u^+ + (Y_{k-1})_u^+ - (Y_{k-1})_u^- & \text{для } u \in X_{k-1} \setminus Z_1, \\ (Y_{k-1})_u^- - (Y_{k-1})_u^+ + (X_{k-1})_u^+ - (X_{k-1})_u^- & \text{для } u \in Y_{k-1} \setminus Z_2. \end{cases}$$

Шаг 2. Выбрать вершину $u_k \in V \setminus (Z_1 \cup Z_2)$ такую, что

$$\delta_k(u_k) = \max_{u \in V \setminus (Z_1 \cup Z_2)} \delta_k(u).$$

Шаг 3. Если $\delta_k(u_k) \leqslant 0$, то **СТОП.** Положить $X' = X_{k-1}$, $Y' = Y_{k-1}$, $M' = M(X', Y')$. **Конец.**
Шаг 4. Если $u_k \in X_{k-1}$, то положить $X_k = X_{k-1} \setminus \{u_k\}$, $Y_k = Y_{k-1} \cup \{u_k\}$. Если же $u_k \in Y_{k-1}$, то положить $X_k = X_{k-1} \cup \{u_k\}$, $Y_k = Y_{k-1} \setminus \{u_k\}$. **Перейти на итерацию k+1**.

Нетрудно заметить, что в отличие от процедуры локального поиска $\mathbf{LS}_{\leqslant \mathbf{2}}$, процедура $\mathbf{LS_2}$ всегда возвращает в качестве решения граф из множества $\mathcal{M}_2(V)$, поскольку вершины $x \in X$ и $y \in Y$ всегда лежат в разных кластерах. Таким образом, данная процедура может быть использована для построения алгоритма приближенного решения задачи $\mathbf{GC_2}$.

Замечание 2.2. *Трудоемкость процедуры* $\mathbf{LS_2}$ – $O(n^3)$.

Доказательство трудоемкости процедуры $\mathbf{LS_2}$ в точности повторяет доказательство трудоемкости процедуры $\mathbf{LS}_{\leqslant \mathbf{3}}$ из раздела 1.1.2.

2.1.4 2-приближенный алгоритм для задачи $\mathbf{GC_2}$

Прежде чем приступить к описанию 2-приближенного алгоритма для задачи $\mathbf{GC_2}$, докажем несколько вспомогательных утверждений.

Лемма 2.2. *Пусть* $M^* = M(X^*, Y^*) \in \mathcal{M}_2(V)$ – *оптимальное решение задачи* $\mathbf{GC_2}$ *на* n*-вершинном графе* $G = (V, E)$, *где* $|X^*| \geqslant 2, |Y^*| \geqslant 2$. *Тогда для любой вершины* $v \in V$ *справедливо неравенство*

$$d_D(v) \leqslant \frac{n}{2},$$

где $D = D(G, M^*)$, $d_D(v)$ – *степень вершины* v *в графе* D.

Доказательство. Предположим противное – существует такая вершина $u \in V$, что $d_D(u) > \frac{n}{2}$, т.е. $d_D(u) = \frac{n}{2} + c$, где $c > 0, \frac{n}{2} + c \in \mathbb{N}$ и $\frac{n}{2} + c \leqslant n - 1$. Рассмотрим граф $\tilde{M} \in \mathcal{M}_2(V)$, полученный из графа M^* путем переноса вершины u в другой кластер. При этом, очевидно, $\tilde{M}$ будет допустимым решением задачи $\mathbf{GC_2}$ на G (поскольку $|X^*| \geqslant 2, |Y^*| \geqslant 2$). Рассмотрим граф $\tilde{D} = D(G, \tilde{M})$. Нетрудно заметить, что степень вершины u в графе $\tilde{D}$ равна «нестепени» вершины u в графе D:

$$d_{\tilde{D}}(u) = n - 1 - \frac{n}{2} - c = \frac{n}{2} - 1 - c.$$

Очевидно, что графы D и $\tilde{D}$ отличаются лишь ребрами вида $up, p \in V$,

остальные ребра у них совпадают. Следовательно,

$$\rho(G,\tilde{M}) - \rho(G,M^*) = |N_{\tilde{D}}(u)| - |N_D(u)| = d_{\tilde{D}}(u) - d_D(u)$$
$$= \frac{n}{2} - 1 - c - \frac{n}{2} - c = -(1+2c) < 0.$$

Из полученного неравенства следует, что M^* не является оптимальным решением, а это противоречит условию.

Лемма 2.2 доказана. □

Если убрать ограничение $|X^*| \geqslant 2, |Y^*| \geqslant 2$, то утверждение леммы 2.2 может стать неверным. Действительно, пусть дан полный граф $K_n, n \geqslant 3$. Оптимальное решение задачи $\mathbf{GC_2}$ на K_n – граф M^*, в котором один кластер состоит из произвольной вершины графа K_n, а другой кластер порожден оставшимися вершинами. Нетрудно заметить, что в графе $D(K_n, M^*)$ степень вершины, образующей отдельный кластер, будет равна $n-1$. При этом степени всех оставшихся вершин все равно будут не больше, чем $\frac{n}{2}$ (это неравенство доказывается так же, как и в лемме 2.2). Однако, воспользовавшись следствием 2.1, мы можем убрать это ограничение из леммы 2.2 и переписать ее в других обозначениях.

Лемма 2.3. *Пусть $G = (V, E)$ – произвольный n-вершинный граф, $n \geqslant 3$, $M^* = M(X^*, Y^*)$ – оптимальное решение задачи $\mathbf{GC_2}$ на G, а M_{vw} – кластерный граф, для которого справедливо следствие 2.1. Тогда для любой вершины $u \in V \setminus \{v, w\}$ справедливы следующие неравенства:*

1. *если $u \in X^*$, то $(X^*)_u^- + (Y^*)_u^+ \leqslant \frac{n}{2}$,*
2. *если $u \in Y^*$, то $(X^*)_u^+ + (Y^*)_u^- \leqslant \frac{n}{2}$.*

Доказательство. Возможны 3 случая.

Случай 1. $|X^*| \geq 2, |Y^*| \geqslant 2$. Доказательство этого случая аналогично доказательству леммы 2.2.

Случай 2. $|X^*| = 1, |Y^*| \geqslant 2$. Тогда в графе M^* кластер X^* состоит из единственной вершины v. Но поскольку утверждение леммы должно быть справедливо для всех вершин $u \in V \setminus \{v, w\}$, то доказательство этого случая также аналогично доказательству леммы 2.2.

Случай 3. $|X^*| \geqslant 2, |Y^*| = 1$. Этот случай доказывается аналогично случаю 2 путем замены вершины v на вершину w.

Лемма 2.3 доказана. □

Рассмотрим следующий алгоритм приближенного решения задачи $\mathbf{GC_2}$.

Алгоритм $\mathbf{NLS_2}$ (Neighborhood with Local Search for $\mathbf{GC_2}$).

Вход: граф $G = (V, E)$.

Выход: кластерный граф $M_{NLS} = M(X, Y) \in \mathcal{M}_2(V)$.

Шаг 1. Пусть F – множество всех допустимых решений, построенных алгоритмом $\mathbf{N_2}$. Применить процедуру локального поиска $\mathbf{LS_2}$ к каждому кластерному графу из множества F.

Шаг 2. Среди всех локальных оптимумов, построенных на шаге 1, выбрать ближайший к G кластерный граф M_{NLS}.

Конец.

Замечание 2.3. *Трудоемкость алгоритма* $\mathbf{NLS_2}$ – $O(n^5)$.

Доказательство трудоемкости алгоритма $\mathbf{NLS_2}$ аналогична доказательству трудоемкости алгоритма $\mathbf{NLS_{\leqslant 3}}$.

К сожалению, метод доказательства гарантированной оценки точности алгоритма **CSW** неприменим для задачи $\mathbf{GC_2}$. Коулман, Саундерсон и Вирт использовали *технику переключений*, позволяющую свести любую индивидуальную задачу $\mathbf{GC_{\leqslant 2}}$ в эквивалентной задаче, оптимальным решением которой является кластерный граф, состоящий из одного кластера. В задаче $\mathbf{GC_2}$ любое оптимальное решение состоит из ровно двух кластеров, поэтому далее будет представлен другой метод доказательства гарантированной оценки точности, не использующий технику переключений.

Для алгоритма $\mathbf{NLS_2}$ справедлива следующая гарантированная оценка точности.

Теорема 2.2. *Для любого графа* $G = (V, E)$ *верно следующее неравенство:*

$$\rho(G, M_{NLS}) \leqslant 2\rho(G, M^*),$$

где $M^* \in \mathcal{M}_2(V)$ *– оптимальное решение задачи* $\mathbf{GC_2}$ *на графе* G, *а* $M_{NLS} \in \mathcal{M}_2(V)$ *– решение, построенное алгоритмом* $\mathbf{NLS_2}$.

Доказательство. Пусть $M^* = M(X^*, Y^*)$ и v – вершина минимальной степени в графе $D = D(G, M^*)$. Согласно следствию 2.1 среди всех графов множества F всегда существует такой граф $M_{vw} = M(X, Y)$, что

1. M_{vw} получен из графа M^* путем переноса не более чем $d_D(v) = d_{\min}$ вершин,
2. $v \in X \cap X^*, w \in Y \cap Y^*$.

Рассмотрим поведение процедуры $\mathbf{LS_2}(\mathbf{M_{vw}}, \mathbf{X}, \mathbf{Y}, \{\mathbf{v}\}, \{\mathbf{w}\})$ на этом графе.

Очевидно, что

$$|X \cap Y^*| \cup |Y \cap X^*| \leqslant d_{\min}.$$

Процедура локального поиска $\mathbf{LS_2}$ начинает свою работу с множеств $X_0 = X$ и $Y_0 = Y$. На каждой итерации k процедура $\mathbf{LS_2}$ либо переносит некоторую вершину $u_k \in V \setminus \{v, w\}$ в другой кластер, либо не переносит ни одну из вершин и заканчивает свою работу.

Рассмотрим итерацию $t+1$, обладающую следующими свойствами:

– на каждой из итераций $k \in \{1, \ldots, t\}$ процедура $\mathbf{LS_2}$ выбирала некоторую вершину $u_k \in (X \cap Y^*) \cup (Y \cap X^*)$;

– на итерации $t+1$ процедура $\mathbf{LS_2}$ либо выбирает вершину $u_{t+1} \in ((X \cap X^*) \cup (Y \cap Y^*)) \setminus \{v, w\}$, либо итерация $t+1$ является последней для процедуры $\mathbf{LS_2}$.

Введем следующие величины:

$$\alpha_{t+1}(u) = \begin{cases} (X_t \cap X^*)_u^- - (X_t \cap X^*)_u^+ + (Y_t \cap Y^*)_u^+ - (Y_t \cap Y^*)_u^-, \ u \in X_t \cap Y^* \\ (Y_t \cap Y^*)_u^- - (Y_t \cap Y^*)_u^+ + (X_t \cap X^*)_u^+ - (X_t \cap X^*)_u^-, \ u \in Y_t \cap X^*, \end{cases}$$

$$\beta_{t+1}(u) = \begin{cases} (X_t \cap Y^*)_u^- - (X_t \cap Y^*)_u^+ + (Y_t \cap X^*)_u^+ - (Y_t \cap X^*)_u^-, \ u \in X_t \cap Y^* \\ (Y_t \cap X^*)_u^- - (Y_t \cap X^*)_u^+ + (X_t \cap Y^*)_u^+ - (X_t \cap Y^*)_u^-, \ u \in Y_t \cap X^*. \end{cases}$$

Тогда для каждой вершины $u \in (X_t \cap Y^*) \cup (Y_t \cap X^*)$ величина $\delta_{t+1}(u)$

(шаг 1 процедуры $\mathbf{LS_2}$) раскладывается в сумму величин $\alpha_{t+1}(u)$ и $\beta_{t+1}(u)$:

$$\delta_{t+1}(u) = \alpha_{t+1}(u) + \beta_{t+1}(u). \tag{2.5}$$

Действительно, если $u \in X_t \cap Y^*$, то

$$\begin{gathered}\delta_{t+1}(u) = (X_t)_u^- - (X_t)_u^+ + (Y_t)_u^+ - (Y_t)_u^- \\ = (X_t \cap X^*)_u^- - (X_t \cap X^*)_u^+ + (Y_t \cap Y^*)_u^+ - (Y_t \cap Y^*)_u^- \\ +(X_t \cap Y^*)_u^- - (X_t \cap Y^*)_u^+ + (Y_t \cap X^*)_u^+ - (Y_t \cap X^*)_u^- \\ = \alpha_{t+1}(u) + \beta_{t+1}(u).\end{gathered}$$

Для вершин $u \in Y_t \cap X^*$ равенство (2.5) доказывается аналогично.

Рассмотрим кластерный граф $M_t = M(X_t, Y_t)$. Согласно лемме 2.1

$$\rho(G, M_t) - \rho(G, M^*) = \sum_{u \in X_t \cap Y^*} \alpha_{t+1}(u) + \sum_{u \in Y_t \cap X^*} \alpha_{t+1}(u).$$

Поскольку на итерациях $k \in \{1, \ldots, t\}$ процедурой $\mathbf{LS_2}$ перемещались только вершины из множества $(X \cap Y^*) \cup (Y \cap X^*)$, то

$$|X_t \cap Y^*| + |Y_t \cap X^*| = r \leqslant d_{\min}. \tag{2.6}$$

Но тогда

$$\rho(G, M_t) - \rho(G, M^*) \leqslant r \max\{\alpha_{t+1}(u) : u \in (X_t \cap Y^*) \cup (Y_t \cap X^*)\}. \tag{2.7}$$

Для каждой вершины $u \in V \setminus \{v, w\}$ оценим величину ее локального улучшения $\delta_{t+1}(u)$ (другими словами, как изменится значение целевой функции при переносе вершины u в другой кластер).

(а) Докажем, что для всех вершин $u \in \Big((X_t \cap X^*) \cup (Y_t \cap Y^*)\Big) \setminus \{v, w\}$ справедливо следующее неравенство:

$$\delta_{t+1}(u) \leqslant 2(|Y_t \cap X^*| + |X_t \cap Y^*|) + 1. \tag{2.8}$$

Приведем доказательство для вершин $u \in (X_t \cap X^*) \setminus \{v\}$. Для вершин $u \in (Y_t \cap Y^*) \setminus \{w\}$ неравенство доказывается аналогично с помошью симмет-

ричной замены X_t и X^* на Y_t и Y^*.

Заметим, что

$$(Y_t \cap X^*)_u^+ + (Y_t \cap X^*)_u^- + (X_t \cap Y^*)_u^+ + (X_t \cap Y^*)_u^- = |Y_t \cap X^*| + |X_t \cap Y^*|, \quad (2.9)$$

$$\begin{aligned}(X_t \cap X^*)_u^+ + (X_t \cap X^*)_u^- + (Y_t \cap Y^*)_u^+ + (Y_t \cap Y^*)_u^- \\ = n - 1 - |Y_t \cap X^*| - |X_t \cap Y^*|.\end{aligned} \quad (2.10)$$

Согласно лемме 2.3

$$(X_t \cap X^*)_u^- + (Y_t \cap Y^*)_u^+ \leq (X^*)_u^- + (Y^*)_u^+ \leqslant \frac{n}{2}. \quad (2.11)$$

По определению, для вершины u из множества $(X_t \cap X^*) \setminus \{v\}$ величина $\delta_{t+1}(u)$ равна

$$\begin{aligned}\delta_{t+1}(u) = (X_t)_u^- - (X_t)_u^+ + (Y_t)_u^+ - (Y_t)_u^- \\ = (X_t \cap X^*)_u^- - (X_t \cap X^*)_u^+ + (Y_t \cap Y^*)_u^+ - (Y_t \cap Y^*)_u^- \\ + (X_t \cap Y^*)_u^- - (X_t \cap Y^*)_u^+ + (Y_t \cap X^*)_u^+ - (Y_t \cap X^*)_u^-.\end{aligned}$$

Добавим и отнимем величину $(X_t \cap X^*)_u^- + (Y_t \cap Y^*)_u^+$. Тогда

$$\begin{aligned}\delta_{t+1}(u) = 2((X_t \cap X^*)_u^- + (Y_t \cap Y^*)_u^+) - (X_t \cap X^*)_u^- - (Y_t \cap Y^*)_u^+ \\ - (X_t \cap X^*)_u^+ - (Y_t \cap Y^*)_u^- + (X_t \cap Y^*)_u^- - (X_t \cap Y^*)_u^+ + (Y_t \cap X^*)_u^+ - (Y_t \cap X^*)_u^-.\end{aligned}$$

Используя (2.10) и (2.11), получим, что

$$\begin{aligned}\delta_{t+1}(u) \leqslant 2\frac{n}{2} - ((X_t \cap X^*)_u^- + (Y_t \cap Y^*)_u^+ + (X_t \cap X^*)_u^+ + (Y_t \cap Y^*)_u^-) \\ + (X_t \cap Y^*)_u^- - (X_t \cap Y^*)_u^+ + (Y_t \cap X^*)_u^+ - (Y_t \cap X^*)_u^- = n - n + 1 \\ + |Y_t \cap X^*| + |X_t \cap Y^*| + (X_t \cap Y^*)_u^- - (X_t \cap Y^*)_u^+ + (Y_t \cap X^*)_u^+ - (Y_t \cap X^*)_u^-.\end{aligned}$$

Поскольку все члены в правой части неравенства неотрицательны, то

$$\begin{aligned}(X_t \cap Y^*)_u^- - (X_t \cap Y^*)_u^+ + (Y_t \cap X^*)_u^+ - (Y_t \cap X^*)_u^- \\ \leqslant (X_t \cap Y^*)_u^- + (X_t \cap Y^*)_u^+ + (Y_t \cap X^*)_u^+ + (Y_t \cap X^*)_u^-.\end{aligned}$$

Тогда, используя (2.9), получим, что

$$\delta_{t+1}(u) \leqslant |Y_t \cap X^*| + |X_t \cap Y^*| + 1 + (X_t \cap Y^*)_u^- + (X_t \cap Y^*)_u^+ +$$
$$+ (Y_t \cap X^*)_u^+ + (Y_t \cap X^*)_u^- \leqslant |Y_t \cap X^*| + |X_t \cap Y^*| + 1 + |Y_t \cap X^*| + |X_t \cap Y^*|$$
$$= 2(|Y_t \cap X^*| + |X_t \cap Y^*|) + 1.$$

(b) Теперь докажем, что для всех вершин $u \in (Y_t \cap X^*) \cup (X_t \cap Y^*)$

$$\delta_{t+1}(u) \leqslant 2(|Y_t \cap X^*| + |X_t \cap Y^*|) + 1. \tag{2.12}$$

Действительно, если итерация $t+1$ является последней итерацией процедуры $\mathbf{LS_2}$, то

$$\delta_{t+1}(u) \leqslant 0 < 2(|Y_t \cap X^*| + |X_t \cap Y^*|) + 1.$$

Если же итерация $t+1$ не является последней, то процедура $\mathbf{LS_2}$ выбрала для переноса некоторую вершину $u_{t+1} \in ((X \cap X^*) \cup (Y \cap Y^*)) \setminus \{v, w\}$. Поскольку на итерациях $k \in \{1, \dots t\}$ процедурой $\mathbf{LS_2}$ перемещались только вершины из $(X \cap Y^*) \cup (Y \cap X^*)$, то $X \cap X^* \subset X_t \cap X^*$ и $Y \cap Y^* \subset Y_t \cap Y^*$. Следовательно, $u_{t+1} \in ((X_t \cap X^*) \cup (Y_t \cap Y^*)) \setminus \{v, w\}$, а значит, согласно (2.8),

$$\delta_{t+1}(u) \leqslant \delta_{t+1}(u_{t+1}) \leqslant 2(|Y_t \cap X^*| + |X_t \cap Y^*|) + 1.$$

Далее докажем, что на итерации $t+1$ для каждой вершины u из множества $(X_t \cap Y^*) \cup (Y_t \cap X^*)$ справедливо следующее неравенство:

$$\alpha_{t+1}(u) \leqslant \frac{n}{2}. \tag{2.13}$$

Приведем доказательство для вершин $u \in Y_t \cap X^*$. Для вершин $u \in X_t \cap Y^*$ неравенство доказывается аналогично с помошью симметричной замены X_t и X^* на Y_t и Y^*.

Предположим противное, т.е. существует такая вершина $p \in Y_t \cap X^*$, что $\alpha_{t+1}(p) > \frac{n}{2}$. Согласно (2.5)

$$\beta_{t+1}(p) = \delta_{t+1}(p) - \alpha_{t+1}(p) < \delta_{t+1}(p) - \frac{n}{2}.$$

Тогда, в силу (2.12), $\delta_{t+1}(p) \leqslant 2(|Y_t \cap X^*|+|X_t \cap Y^*|) + 1$, а значит,

$$\beta_{t+1}(p) < 2(|Y_t \cap X^*|+|X_t \cap Y^*|) + 1 - \frac{n}{2}. \tag{2.14}$$

Поскольку $d_{\min}$ – наименьшая из степеней графа $D = D(G, M^*)$, то

$$d_D(p) = (Y_t \cap X^*)_p^- + (X_t \cap X^*)_p^- + (X_t \cap Y^*)_p^+ + (Y_t \cap Y^*)_p^+ \geqslant d_{\min}.$$

Тогда, используя (2.6), мы получим, что

$$(Y_t \cap X^*)_p^- + (X_t \cap X^*)_p^- + (X_t \cap Y^*)_p^+ + (Y_t \cap Y^*)_p^+ \geqslant |Y_t \cap X^*|+|X_t \cap Y^*|. \tag{2.15}$$

Поскольку $p \in Y_t \cap X^*$, то

$$\begin{aligned}(Y_t \cap X^*)_p^+ + (Y_t \cap X^*)_p^- + (X_t \cap Y^*)_p^+ + (X_t \cap Y^*)_p^- \\ = |Y_t \cap X^*|+|X_t \cap Y^*|-1,\end{aligned} \tag{2.16}$$

$$\begin{aligned}(X_t \cap X^*)_p^+ + (X_t \cap X^*)_p^- + (Y_t \cap Y^*)_p^+ + (Y_t \cap Y^*)_p^- \\ = n - |Y_t \cap X^*|-|X_t \cap Y^*|.\end{aligned} \tag{2.17}$$

Используя (2.16), мы получим, что

$$\begin{aligned}\beta_{t+1}(p) = (Y_t \cap X^*)_p^- - (Y_t \cap X^*)_p^+ + (X_t \cap Y^*)_p^+ - (X_t \cap Y^*)_p^- \\ = (Y_t \cap X^*)_p^- + (X_t \cap Y^*)_p^+ + (Y_t \cap X^*)_p^- + (X_t \cap Y^*)_p^+ - |Y_t \cap X^*|-|X_t \cap Y^*|+1 \\ = 2((Y_t \cap X^*)_p^- + (X_t \cap Y^*)_p^+) - |Y_t \cap X^*|-|X_t \cap Y^*|+1.\end{aligned}$$

Из (2.15) следует, что

$$(Y_t \cap X^*)_p^- + (X_t \cap Y^*)_p^+ \geqslant |Y_t \cap X^*|+|X_t \cap Y^*|-(X_t \cap X^*)_p^- - (Y_t \cap Y^*)_p^+,$$

поэтому

$$\begin{aligned}\beta_{t+1}(p) \geqslant 2(|Y_t \cap X^*|+|X_t \cap Y^*|-(X_t \cap X^*)_p^- - (Y_t \cap Y^*)_p^+) - |Y_t \cap X^*| \\ -|X_t \cap Y^*|+1 = |Y_t \cap X^*|+|X_t \cap Y^*|+1 - 2(X_t \cap X^*)_p^- - 2(Y_t \cap Y^*)_p^+.\end{aligned}$$

Добавим и отнимем величину $(X_t \cap X^*)_p^+ + (Y_t \cap Y^*)_p^-$. Тогда

$$\beta_{t+1}(p) \geqslant |Y_t \cap X^*| + |X_t \cap Y^*| + 1 + (Y_t \cap Y^*)_p^- - (Y_t \cap Y^*)_p^+ + (X_t \cap X^*)_p^+$$
$$-(X_t \cap X^*)_p^- - ((Y_t \cap Y^*)_p^- + (Y_t \cap Y^*)_p^+ + (X_t \cap X^*)_p^+ + (X_t \cap X^*)_p^-).$$

Теперь, используя (2.17), мы получим, что

$$\beta_{t+1}(p) \geqslant |Y_t \cap X^*| + |X_t \cap Y^*| + 1 + (Y_t \cap Y^*)_p^- - (Y_t \cap Y^*)_p^+ + (X_t \cap X^*)_p^+$$
$$-(X_t \cap X^*)_p^- - n + |Y_t \cap X^*| + |X_t \cap Y^*| = 2(|Y_t \cap X^*| + |X_t \cap Y^*|)$$
$$+1 - n + (Y_t \cap Y^*)_p^- - (Y_t \cap Y^*)_p^+ + (X_t \cap X^*)_p^+ - (X_t \cap X^*)_p^-.$$

Остается заметить, что поскольку $p \in Y_t \cap X^*$, то

$$\alpha_{t+1}(p) = (Y_t \cap Y^*)_p^- - (Y_t \cap Y^*)_p^+ + (X_t \cap X^*)_p^+ - (X_t \cap X^*)_p^-,$$

а значит,

$$\beta_{t+1}(p) \geqslant 2(|Y_t \cap X^*| + |X_t \cap Y^*|) + 1 - n + \alpha_{t+1}(p) >$$
$$2(|Y_t \cap X^*| + |X_t \cap Y^*|) + 1 - \frac{n}{2}.$$

Последнее неравенство противоречит неравенству (2.14). Значит, в силу произвольности вершины p, для каждой вершины $u \in Y_t \cap X^*$ имеет место неравенство (2.13).

Теперь, используя выражения (2.6), (2.7), (2.13), а также лемму 1.1, получим

$$\rho(G, M'_{vw}) - \rho(G, M^*) \leqslant \rho(G, M_t) - \rho(G, M^*)$$
$$\leqslant r \max\{\alpha_{t+1}(u) : u \in (X_t \cap Y^*) \cup (Y_t \cap X^*)\} \leqslant r\frac{n}{2} \leqslant d_{\min}\frac{n}{2} \leqslant \rho(G, M^*).$$

Значит,

$$\rho(G, M'_{vw}) \leqslant 2\rho(G, M^*).$$

Граф M'_{vw} будет построен на шаге 1 алгоритма $\mathbf{NLS_2}$, откуда и следует гарантированная оценка точности алгоритма $\mathbf{NLS_2}$.

Теорема 2.2 доказана. □

2.2 Задачи и методы кластеризации с частичным обучением

В этом параграфе будут рассмотрены задачи и методы кластеризации с частичным обучением. В случае, когда число кластеров равно 2, по аналогии с задачей $\mathbf{GC_2}$ будут представлены два полиномиальных приближенных алгоритма с гарантированными оценками точности.

2.2.1 Постановки задач и вспомогательные утверждения

Рассмотрим следующие варианты задач $\mathbf{SGC_k}$ и $\mathbf{SSGC_k}$ при $k = 2$.

Задача $\mathbf{SGC_2}$ (2-SEMI-SUPERVISED GRAPH CLUSTERING). Дан обыкновенный граф $G = (V, E)$ и произвольное множество $Z = \{z_1, z_2\} \subset V$. Требуется найти такой граф $M^* \in \mathcal{M}_k(V)$, что

$$\rho(G, M^*) = \min_{M \in \mathcal{M}_2(V)} \rho(G, M),$$

причем минимум берется по всем кластерным графам $M = (V, E_M) \in \mathcal{M}_k(V)$, в которых $z_1 z_2 \notin E_M$.

Задача $\mathbf{SSGC_2}$ (2-SET SEMI-SUPERVISED GRAPH CLUSTERING). Дан обыкновенный граф $G = (V, E)$ и семейство $\mathcal{Z} = \{Z_1, Z_2\}$ непересекающихся непустых подмножеств множества V. Требуется найти такой граф $M^* \in \mathcal{M}_k(V)$, что

$$\rho(G, M^*) = \min_{M \in \mathcal{M}_k(V)} \rho(G, M),$$

причем минимум берется по всем кластерным графам $M = (V, E_M) \in \mathcal{M}_k(V)$, в которых

1. $zz' \notin E_M$ для всех $z \in Z_1, z' \in Z_2$;
2. $zz' \in E_M$ для всех $z, z' \in Z_i, i = 1, 2$.

Очевидно, что задача $\mathbf{SGC_2}$ является частным случаем задачи $\mathbf{SSGC_2}$ при $|Z_1| = |Z_2| = 1$.

Следующая лемма является аналогом леммы 2.3 для задач $\mathbf{SGC_2}$ и $\mathbf{SSGC_2}$ и доказывается аналогично.

Лемма 2.4. *Пусть* $M^* = M(X^*, Y^*) \in \mathcal{M}_2(V)$ *– оптимальное решение либо задачи* $\mathbf{SGC_2}$*, либо задачи* $\mathbf{SSGC_2}$ *на* n*-вершинном графе* $G = (V, E), \mathcal{Z} = \{Z_1, Z_2\}$*. Тогда для любой вершины* $v \in V \setminus (Z_1 \cup Z_2)$ *справедливы неравенства:*

1. *если* $v \in X^*$*, то* $(X^*)_v^- + (Y^*)_v^+ \leq \frac{n}{2}$,
2. *если* $v \in Y^*$*, то* $(X^*)_v^+ + (Y^*)_v^- \leq \frac{n}{2}$.

2.2.2 Приближенные алгоритмы для задач $\mathbf{SGC_2}$ и $\mathbf{SSGC_2}$

В этом параграфе мы покажем, что по аналогии с задачей $\mathbf{GC_2}$ можно построить два приближенных алгоритма для задач $\mathbf{SGC_2}$ и $\mathbf{SSGC_2}$. Первый алгоритм является аналогом алгоритма $\mathbf{N_2}$ для задачи $\mathbf{GC_2}$.

Алгоритм $\mathbf{NS_2}$ (Neighborhood semi-supervised for $\mathbf{SGC_2}$ and $\mathbf{SSGC_2}$).

Вход: граф $G = (V, E), Z_1, Z_2$ – непустые непересекающиеся подмножества множества V.

Выход: кластерный граф $M_{NS} = M(X, Y) \in \mathcal{M}_2(V)$, Z_1, Z_2 подмножества разных кластеров.

Шаг 1. Для каждой вершины $v \in V$ выполнить:

(а) Если $v \notin Z_1 \cup Z_2$, то построить два кластерных графа $\overline{M}_v = M(\overline{X}, \overline{Y})$ и $\overline{\overline{M}}_v = M(\overline{\overline{X}}, \overline{\overline{Y}})$, где

$$\overline{X} = \{v\} \cup ((N_G(v) \cup Z_1) \setminus Z_2), \overline{Y} = V \setminus \overline{X},$$
$$\overline{\overline{X}} = \{v\} \cup ((N_G(v) \cup Z_2) \setminus Z_1), \overline{\overline{Y}} = V \setminus \overline{\overline{X}}.$$

(б) Если $v \in Z_1 \cup Z_2$, то построить граф $M_v = M(X, Y)$, где

$$X = \{v\} \cup ((N_G(v) \cup Z) \setminus \overline{Z}), Y = V \setminus X.$$

Здесь $Z = Z_1, \overline{Z} = Z_2$ если $v \in Z_1$, или $Z = Z_2, \overline{Z} = Z_1$ если $v \in Z_2$.

Шаг 2. Среди всех кластерных графов, построенных на шаге 1, выбрать ближайший к G кластерный граф $M_{NS} = M(X, Y)$.

Конец.

Замечание 2.4. *Трудоемкость алгоритма* $\mathbf{NS_2}$ – $O(n^2)$.

Доказательство трудоемкости алгоритма $\mathbf{NS_2}$ аналогична доказательству трудоемкости алгоритма **BBC** в разделе 1.1.2.

Для алгоритма $\mathbf{NS_2}$ справедлива следующая гарантированная оценка точности.

Теорема 2.3. *Для любого графа* $G = (V, E)$ *и для произвольных непустых непересекающихся множеств* $Z_1, Z_2 \subset V$ *имеет место неравенство*

$$\rho(G, M_{NS}) \leqslant 3\rho(G, M^*),$$

где $M^* \in \mathcal{M}_2(V)$ *– оптимальное решение задачи* $\mathbf{SGC_2}$ *или* $\mathbf{SSGC_2}$ *на графе* G, *а* M_{NS} *– кластерный граф, построенный алгоритмом* $\mathbf{NS_2}$.

Доказательство. Пусть v - вершина минимальной степени в графе $D = D(G, M^*)$, т.е. $d_D(v) = d_{\min}$. Без ограничения общности будем считать, что $v \in X \cap X^*$. Докажем, что на шаге 1 алгоритма $\mathbf{NS_2}$ будет построен кластерный граф $M = M(X, Y)$, полученный из графа M^* путем переноса не более $d_{\min}$ вершин.

По определению графа D

$$X^* = \{v\} \cup (N_G(v) \setminus N_D(v)) \cup (\overline{N}_G(v) \cap N_D(v)), \tag{2.18}$$

$$N_G(v) = (N_G(v) \setminus N_D(v)) \cup (N_G(v) \cap N_D(v)). \tag{2.19}$$

Заметим, что возможны 4 случая.

1. $v \notin Z_1 \cup Z_2, Z_1 \subset X^*, Z_2 \subset Y^*$;
2. $v \notin Z_1 \cup Z_2, Z_2 \subset X^*, Z_1 \subset Y^*$;
3. $v \in Z_1, Z_1 \subset X^*, Z_2 \subset Y^*$;
4. $v \in Z_2, Z_1 \subset Y^*, Z_2 \subset X^*$.

Докажем утверждение теоремы в случае 1. Доказательство случая 3 эквивалентно доказательству случая 1. Доказательство случаев 2 и 4 получается из доказательства случая 1 путем симметричной замены множеств Z_1 и Z_2.

Итак, пусть $v \notin Z_1 \cup Z_2, Z_1 \subset X^*, Z_2 \subset Y^*$. Тогда согласно пункту (а) шага 1 в качестве $M = M(X,Y)$ можно взять кластерный граф $\overline{M}_v = M(\overline{X}, \overline{Y})$, где

$$\overline{X} = \{v\} \cup ((N_G(v) \cup \{Z_1\}) \setminus \{Z_2\}), \overline{Y} = V \setminus \overline{X}.$$

Вычислим мощность множества $X \Delta X^*$. Для это введем следующие множества: $Z_{11} = Z_1 \cap N_G(v)$, $Z_{12} = Z_1 \cap \overline{N}_G(v)$, $Z_{21} = Z_2 \cap N_G(v)$, $Z_{22} = Z_2 \cap \overline{N}_G(v)$. Легко видеть, что

$$Z_{21} \subseteq N_G(v) \cap N_D(v), \tag{2.20}$$

$$Z_{12} \subseteq \overline{N}_G(v) \cap N_D(v). \tag{2.21}$$

Используя (2.19), (2.20) и (2.21), получим

$$\begin{aligned} X &= \{v\} \cup ((N_G(v) \cup Z_1) \setminus Z_2) = \{v\} \cup ((N_G(v) \cup Z_{12}) \setminus Z_{21}) \\ &= \{v\} \cup \Big(((N_G(v) \setminus N_D(v)) \cup (\overline{N}_G(v) \cap N_D(v)) \cup Z_{12}) \setminus Z_{21}\Big) \\ &= \{v\} \cup (N_G(v) \setminus N_D(v)) \cup ((\overline{N}_G(v) \cap N_D(v)) \setminus Z_{21}) \cup Z_{12}. \end{aligned}$$

Теперь, используя (2.18), получим

$$\begin{aligned} X^* \Delta X &= ((N_G(v) \cap N_D(v)) \cup (\overline{N}_G(v) \cap N_D(v))) \setminus (Z_{12} \cup Z_{21}) \\ &= N_D(v) \setminus (Z_{12} \cup Z_{21}). \end{aligned}$$

Следовательно,

$$|X^* \Delta X| = |N_D(v) \setminus (Z_{12} \cup Z_{21})| = |N_D(v)| - (|Z_{12}| + |Z_{21}|) \leqslant |N_D(v)| = d_{\min}.$$

Итак, мы показали, что на шаге 1 алгоритма $\mathbf{NS_2}$ будет построен кластерный граф M, полученный из графа M^* путем переноса не более чем $d_{\min}$ вершин в другой кластер. Как было отмечено ранее, при переносе $d_{\min}$ вершин значение целевой функции не может увеличиться более чем на $nd_{\min}$, так как перенос одной вершины увеличивает значение целевой функции не более чем на $n = |V|$. Отсюда с учетом леммы 1.1 получаем

$$\rho(G, M) \leqslant \rho(G, M^*) + nd_{\min} \leqslant \rho(G, M^*) + 2\rho(G, M^*) = 3\rho(G, M^*).$$

Теорема 2.3 доказана. □

Следствие 2.2. *Среди всех графов, построенных алгоритмом* $\mathbf{NS_2}$ *на шаге 1, всегда существует такой кластерный граф* $M = M(X, Y)$, *что*

1. M *может быть получен из* M^* *путем переноса не более чем* $d_D(v) = d_{\min}$ *вершин в другой кластер (здесь* $D = D(G, M^*)$*);*
2. *Если* $Z_1 \subset X^*, Z_2 \subset Y^*$, *то* $Z_1 \subset X \cap X^*, Z_2 \subset Y \cap Y^*$. *Иначе если* $Z_2 \subset X^*, Z_1 \subset Y^*$, *то* $Z_1 \subset Y \cap Y^*, Z_2 \subset X \cap X^*$.

Теперь мы можем описать 2-приближенный алгоритм решения задач $\mathbf{SGC_2}$ и $\mathbf{SSGC_2}$.

Алгоритм $\mathbf{NSLS_2}$ (Neighborhood semi-supervised with Local Search for $\mathbf{SGC_2}$ и $\mathbf{SSGC_2}$).

Вход: граф $G = (V, E), Z_1, Z_2$ – непустые непересекающиеся подмножества множества V.
Выход: кластерный граф $M_{NS} = M(X, Y) \in \mathcal{M}_2(V)$, Z_1, Z_2 подмножества разных кластеров.

Шаг 1. Пусть F – множество всех допустимых решений, построенных алгоритмом $\mathbf{NS_2}$. Применить процедуру $\mathbf{LS_2}$ к каждому кластерному графу из множества F.
Шаг 2. Среди всех локальных оптимумов, построенных на шаге 1, выбрать ближайший к G кластерный граф $M_{NSLS} = M(X, Y)$.
Конец.

Замечание 2.5. *Трудоемкость алгоритма* $\mathbf{NSLS_2}$ – $O(n^4)$.

Доказательство трудоемкости алгоритма $\mathbf{NSLS_2}$ аналогична доказательству трудоемкости алгоритма **CSW** в разделе 1.1.2.

Из леммы 2.4 и следствия 2.2 следует гарантированная оценка точности алгоритма $\mathbf{NSLS_2}$.

Теорема 2.4. *Для любого графа $G = (V, E)$ и для произвольных непустых непересекающихся множеств $Z_1, Z_2 \subset V$ имеет место неравенство*

$$\rho(G, M_{NSLS}) \leqslant 2\rho(G, M^*),$$

где $M^* \in \mathcal{M}_2(V)$ – оптимальное решение задачи $\mathbf{SGC_2}$ *или* $\mathbf{SSGC_2}$ *на графе G, а M_{NSLS} – кластерный граф, построенный алгоритмом* $\mathbf{NSLS_2}$.

Доказательство теоремы 2.4 аналогично доказательству теоремы 2.2, только вместо леммы 2.3 и следствия 2.1 используется лемма 2.4 и следствие 2.2.

2.3 Взаимосвязь задач $\mathbf{GC_2}$, $\mathbf{SGC_2}$ и $\mathbf{SSGC_2}$

В данном параграфе мы рассмотрим взаимосвязь задач $\mathbf{GC_2}$, $\mathbf{SGC_2}$ и $\mathbf{SSGC_2}$.

Как было сказано ранее, задача $\mathbf{SGC_k}$ является частным случаем задачи $\mathbf{SSGC_k}$ в случае $|Z_1| = \cdots = |Z_k| = 1$.

Заметим следующее. Если мы решим задачу $\mathbf{SGC_2}$ для каждой упорядоченной пары вершин $(v, w) \in V \times V$ некоторого графа $G = (V, E)$, мы решим задачу $\mathbf{GC_2}$ на этом графе. Очевидно, что среди всех пар вершин графа G всегда существует пара (v, w) такая, что $v \in X^*$, $w \in Y^*$ (здесь $M^* = M(X^*, Y^*)$ – оптимальное решение задачи $\mathbf{GC_2}$ на графе G). Пользуясь этим наблюдением, мы можем построить еще один алгоритм приближенного решения задачи $\mathbf{GC_2}$.

Алгоритм $\mathbf{NSLS_2^*}$ (Neighborhood semi-supervised with Local Search for $\mathbf{GC_2}$).

Вход: граф $G = (V, E)$.
Выход: граф $M_{NSLS^*} \in \mathcal{M}_2(V)$.

Шаг 1. Для каждой упорядоченной пары вершин $(v, w) \in V \times V, v \neq w$ приближенно решить задачу $\mathbf{SGC_2}$ на графе G и множестве $Z = \{v, w\}$ алгоритмом $\mathbf{NSLS_2}$. Полученное решение обозначить через M_{vw}.
Step 2. Среди всех допустимы решений, построенных на шаге 1, выбрать ближайший к G кластерный граф $M_{NSLS^*} \in \mathcal{M}_2(V)$.
Конец.

Для алгоритма $\mathbf{NSLS_2^*}$ справедлива следующая гарантированная оценка точности.

Теорема 2.5. *Для произвольного графа $G = (V, E)$ имеет место неравенство*

$$\rho(G, M_{NSLS^*}) \leqslant 2\rho(G, M^*),$$

где M^* – оптимальное решение задачи $\mathbf{GC_2}$ на графе G, а $M_{NSLS^*} \in \mathcal{M}_2(V)$ – кластерный граф, построенный алгоритмом $\mathbf{NSLS_2^*}$.

Заметим, что алгоритм $\mathbf{NSLS_2^*}$ вызывает $n(n-1)$ раз алгоритм $\mathbf{NSLS_2}$, что значительно увеличивает его трудоемкость. Алгоритм $\mathbf{NSLS_2^*}$ имеет такую же гарантированную оценку точности, что и алгоритм $\mathbf{NLS_2}$, поэтому он может быть интересен лишь с теоретической точки зрения и его использование в практических целях является нецелесообразным.

Глава 3

Точные методы и экспериментальное исследование задач кластеризации вершин графа

В данной главе будут рассмотрены методы нахождения точных решений различных вариантов задачи кластеризации вершин графа, а также представлены результаты экспериментального исследования точных и приближенных алгоритмов.

Предложены два подхода к нахождению точных решений задач кластеризации вершин графа. Проведено экспериментальное исследование точных и приближенных алгоритмов. Цель исследования состояла в том, чтобы на основе статистических данных оценить практическую трудоемкость и точность изучаемых алгоритмов и, используя полученные оценки, сравнить алгоритмы между собой.

3.1 Точные методы для задач кластеризации вершин графа

В этом параграфе будут предложены два подхода к нахождения точных решений различных вариантов задач кластеризации вершин графа. Первый подход основан на универсальном методе ветвей и границ, который может находить оптимальное решение для любой из рассматриваемых задач. Второй подход опирается на модели целочисленного линейного программирования для некоторых вариантов рассматриваемых задач.

3.1.1 Постановки задач и вспомогательные утверждения

Напомним, что если $G_1 = (V, E_1)$ и $G_2 = (V, E_2)$ – обыкновенные помеченные графы на одном и том же множестве вершин V, то расстояние $\rho(G_1, G_2)$ между ними определяется как

$$\rho(G_1, G_2) = |E_1 \Delta E_2| = |E_1 \setminus E_2| + |E_2 \setminus E_1|,$$

т.е. $\rho(G_1, G_2)$ – это число несовпадающих ребер в графах G_1 и G_2.

Утверждение 3.1. *$\rho(G_1, G_2)$ удовлетворяет неравенству*

$$0 \leqslant \rho(G_1, G_2) \leqslant \frac{n(n-1)}{2}.$$

Доказательство. Нетрудно заметить, что если графы G_1 и G_2 совпадают, то $\rho(G_1, G_2) = 0$. Максимум значения $\rho(G_1, G_2)$ достигается в случае, когда $G_1 = O_n$ – пустой n-вершинный граф, а $G_2 = K_n$ – полный n-вершинный граф, т.е. в значении расстояния учитываются все возможные $\frac{n(n-1)}{2}$ ребра.

Утверждение 3.1 доказано. □

Для произвольного графа $G = (V, E)$ и разбиения $(V_1, \ldots, V_k)$ множества вершин V определим следующую функцию:

$$f_k(V_1, \ldots, V_k) = |\{uv \in E : u \in V_i, v \in V_j, i, j \in \{1, \ldots, k\}, i < j\}| \qquad (3.1)$$
$$+|\{uv \notin E : u, v \in V_i, i \in \{1, \ldots, k\}\}|.$$

Нетрудно заметить, что $f_k(V_1, \ldots, V_k) = \rho(G, M)$, где $M = M(V_1, \ldots, V_k)$ – кластерный граф, порожденный множествами вершин $V_1, \ldots, V_k$.

Введя функцию $f_k(V_1, \ldots, V_k)$, мы можем рассмотреть следующие эквивалентные постановки задач **GC**, $\mathbf{GC_k}$, $\mathbf{GC_{\leqslant k}}$, $\mathbf{SGC_k}$ и $\mathbf{SSGC_k}$.

Задача GC (GRAPH CLUSTERING). Для произвольного графа $G = (V, E)$ найти разбиение $(V_1, \ldots, V_n)$ множества V, $|V| = n$, на котором достигает минимума функция $f_n(V_1, \ldots, V_n)$. При этом некоторые из множеств $V_1, \ldots, V_n$ могут быть пустыми.

Задача $\mathbf{GC_{\leqslant k}}$ ([k]-GRAPH CLUSTERING). Для произвольного графа $G = (V, E)$ и целого числа $k, 2 \leqslant k \leqslant |V|$, найти разбиение $(V_1, \ldots, V_k)$

множества V, на котором достигает минимума функция $f_k(V_1, \ldots, V_k)$. При этом некоторые из множеств $V_1, \ldots, V_k$ могут быть пустыми.

Задача $\mathbf{GC_k}$ (k-GRAPH CLUSTERING). Для произвольного графа $G = (V, E)$ и целого числа $k, 2 \leqslant k \leqslant |V|$, найти разбиение $(V_1, \ldots, V_k)$ множества V, на котором достигает минимума функция $f_k(V_1, \ldots, V_k)$. При этом никакие из множеств $V_1, \ldots, V_k$ не могут быть пустыми.

Задача $\mathbf{SGC_k}$ (k-SEMI-SUPERVISED GRAPH CLUSTERING). Для произвольного графа $G = (V, E)$, целого числа $k, 2 \leqslant k \leqslant |V|$, и множества $Z = \{z_1, \ldots, z_k\} (z_i \neq z_j$ при $i \neq j)$, найти разбиение $(V_1, \ldots, V_k)$ множества V, на котором достигает минимума функция $f_k(V_1, \ldots, V_k)$. При этом никакие из множеств $V_1, \ldots, V_k$ не могут быть пустыми и $z_1 \in V_1, \ldots, z_k \in V_k$.

Задача $\mathbf{SSGC_k}$ (k-SET SEMI-SUPERVISED GRAPH CLUSTERING). Для произвольного графа $G = (V, E)$, целого числа $k, 2 \leqslant k \leqslant |V|$, и семейства $\mathcal{Z} = \{Z_1, \ldots, Z_k\}$ попарно непересекающихся непустых подмножеств множества V, найти разбиение $(V_1, \ldots, V_k)$ множества V, на котором достигает минимума функция $f_k(V_1, \ldots, V_k)$. При этом никакие из множеств $V_1, \ldots, V_k$ не могут быть пустыми и $Z_1 \subseteq V1, \ldots, Z_k \subseteq V_k$.

3.1.2 Метод ветвей и границ

Рассмотрим следующую универсальную схему метода ветвей и границ, которая может быть использована для нахождения оптимального решения любой из задач **GC**, $\mathbf{GC_k}$, $\mathbf{GC_{\leqslant k}}$, $\mathbf{SGC_k}$ и $\mathbf{SSGC_k}$ для произвольного графа $G = (V, E), |V| = n$.

Алгоритм $\mathbf{BBM(G, (I_1, \ldots, I_k))}$.

Шаг 1. Положить $S_1 = I_1, \ldots, S_k = I_k$; // $k = n$ для задачи **GC**

Шаг 2. $record = \frac{n(n-1)}{2}$;

Шаг 3. $A = \{j : S_j \neq \emptyset\}$; // множество индексов непустых кластеров

Шаг 4. $B = \{1, \ldots, k\} \setminus A$; // множество индексов пустых кластеров

Шаг 5. $\mathbf{Branch((S_1, \ldots, S_k), record, A, B)}$.

Конец.

Алгоритм ветвей и границ использует следующую процедуру ветвления **Branch**.

Процедура Branch$((\mathbf{S_1}, \ldots, \mathbf{S_k}), \mathbf{record}, \mathbf{A}, \mathbf{B})$.

Если $S_1 \cup \ldots \cup S_k \neq V$:

Шаг 1. Выбрать $v \notin V \setminus (S_1 \cup \ldots \cup S_k)$;

Шаг 2. Для каждого $i \in A$:

Шаг 2.1. $b = \mathbf{Bound}((\mathbf{S_1}, \ldots, \mathbf{S_i} \cup \{\mathbf{v}\}, \ldots, \mathbf{S_k})$;

Шаг 2.2. Если $(b < record)$

$\mathbf{Branch}((\mathbf{S_1}, \ldots, \mathbf{S_i} \cup \{\mathbf{v}\}, \ldots, \mathbf{S_k}), \mathbf{record}, \mathbf{A}, \mathbf{B})$;

Шаг 3. Если $B \neq \emptyset$: // выбрать любое пустое множество S_i

Шаг 3.1. Взять произвольный $i \in B$;

Шаг 3.2. $b = \mathbf{Bound}((\mathbf{S_1}, \ldots, \mathbf{S_i} \cup \{\mathbf{v}\}, \ldots, \mathbf{S_k})$;

Шаг 3.3. Если $(b < record)$

$\mathbf{Branch}((\mathbf{S_1}, \ldots, \mathbf{S_i} \cup \{\mathbf{v}\}, \ldots, \mathbf{S_k}), \mathbf{record}, \mathbf{A}, \mathbf{B})$;

иначе

Шаг 4. $b = \mathbf{Bound}((\mathbf{S_1}, \ldots, \mathbf{S_k})$;

Шаг 5. Если $(\mathbf{updateRecord}(\mathbf{record}, \mathbf{b}, (\mathbf{S_1}, \ldots, \mathbf{S_k})))$ $record = b$.

Конец.

Комментарии.

На вход алгоритму **BBM** передается произвольный граф $G = (V, E)$ и начальное разбиение $(I_1, \ldots, I_k)$ некоторого подмножества множества вершин этого графа. Для задач **GC**, $\mathbf{GC_k}$ и $\mathbf{GC_{\leqslant k}}$ мы можем определить начальное разбиение следующим образом: $I_1 = \emptyset, \ldots, I_k = \emptyset$. Для задачи $\mathbf{SGC_k}$ положим $I_1 = \{z_1\}, \ldots, I_k = \{z_k\}$, а для задачи $\mathbf{SSGC_k}$ $I_1 = Z_1, \ldots, I_k = Z_k$. Воспользовавшись утверждением 3.1, возьмем в качестве начального значения рекорда (параметр *record*) верхнюю оценку значения целевой функции.

Замечание 3.1. *В качестве начального рекорда для целевой функции можно передавать значение, найденное некоторым приближенным алгоритмом.*

Основным элементом алгоритма **BBM** является рекурсивный вызов процедуры **Branch**, реализующий стратегию «ветвления в глубину». Каждый рекурсивный вызов процедуры **Branch** состоит из следующих шагов: выбор очередной вершины для ветвления (шаг 1 процедуры **Branch**), разбиение семейства допустимых решений относительно выбранной вершины (шаг 2 процедуры **Branch**), отсечение неперспективных множеств ветвления (шаг 3 процедуры **Branch**), построение нижней оценки целевой функции для каждого элемента семейства допустимых решений в функции **Bound** (шаги 2 – 4 процедуры **Branch**) и обновление рекорда в функции **updateRecord** (шаг 5 процедуры **Branch**). Рассмотрим подробнее каждый из шагов.

Разбиение семейства допустимых решений и отсечение неперспективных множеств ветвления.

При рекурсивном вызове процедуры **Branch** ей на вход передается разбиение $(S_1, \ldots, S_k)$ подмножества множества вершин графа $G = (V, E)$, определяющее семейство $D(S_1, \ldots, S_k)$ допустимых решений исходной задачи по следующему правилу:

$$D(S_1, \ldots, S_k) = \{(V_1, \ldots, V_k) : S_1 \subseteq V_1, \ldots, S_k \subseteq V_k, V_1 \cup \ldots \cup V_k = V, \\ V_i \cap V_j = \emptyset\}$$

При этом выбранная вершина $v \in V \setminus (S_1 \cup \ldots \cup S_k)$ будет разбивать семейство $D(S_1, \ldots, S_k)$ на k попарно непересекающихся подсемейств. Покажем, что среди этих подсемейств могут быть эквивалентные. Для этого докажем следующее утверждение.

Утверждение 3.2. *Пусть* $B = \{i_1, \ldots, i_m\}$ *и* $|B| = m \geqslant 2$, *т.е. разбиение* $(S_1, \ldots, S_k)$ *содержит* m *пустых множеств* $S_{i_1} = \cdots = S_{i_m} = \emptyset$. *Тогда семейства*

$$D(S_1, \ldots, S_{i_1} \cup \{v\}, \ldots, S_k), \ldots, D(S_1, \ldots, S_{i_m} \cup \{v\}, \ldots, S_k)$$

будут отличаться лишь перестановкой индексов $\{i_1, \ldots, i_m\}$.

Доказательство. Рассмотрим 2 случая.

Случай 1. Пусть $m = 2, B = \{q, r\}$. Рассмотрим перестановку π индексов множеств частичного разбиения $(S_1, \ldots, S_k)$, заданную по правилу:

$$\pi(i) = \begin{cases} i, i \in A \\ r, i = q. \\ q, i = r \end{cases}$$

Другими словами, перестановка $\pi = (q, r)$ – транспозиция, меняющая местами индексы q и r. Пусть далее

$$D(S_1, \ldots, S_q \cup \{v\}, \ldots, S_r, \ldots, S_k)$$
$$= \{(V_1, \ldots, V_k) : S_1 \subseteq V_1, \ldots, S_k \subseteq V_k, V_1 \cup \ldots \cup V_k = V, V_i \cap V_j = \emptyset\},$$
$$D(S_1, \ldots, S_q, \ldots, S_r \cup \{v\}, \ldots, S_k)$$
$$= \{(U_1, \ldots, U_k) : S_1 \subseteq U_1, \ldots, S_k \subseteq U_k, U_1 \cup \ldots \cup U_k = V, U_i \cap U_j = \emptyset\}.$$

Поскольку множества $S_q = S_r = \emptyset$ равны, то каждому конечному разбиению $(V_1, \ldots, V_k)$ множества V взаимно однозначно соответствует конечное разбиение $(U_1, \ldots, U_k)$, отличающееся лишь перестановкой индексов q и r:

$$V_1 = U_1, \ldots, V_q = U_r, \ldots, V_r = U_q, \ldots, V_k = U_k.$$

Следовательно,

$$D(S_1, \ldots, S_q \cup \{v\}, \ldots, S_r, \ldots, S_k)$$
$$= \{(V_1, \ldots, V_k) : S_1 \subseteq V_1, \ldots, S_k \subseteq V_k, V_1 \cup \ldots \cup V_k = V, V_i \cap V_j = \emptyset\}$$
$$= \{(U_{\pi(1)}, \ldots, U_{\pi(k)}) : S_{\pi(1)} \subseteq U_{\pi(1)}, \ldots, S_{\pi(k)} \subseteq U_{\pi(k)}, U_{\pi(1)} \cup \ldots \cup U_{\pi(k)}$$
$$= V, U_{\pi(i)} \cap U_{\pi(j)} = \emptyset\} = D(S_{\pi(1)}, \ldots, S_{\pi(q)}, \ldots, S_{\pi(r)} \cup \{v\}, \ldots, S_{\pi(k)}).$$

Мы показали, что семейства

$$D(S_1, \ldots, S_{i_1} \cup \{v\}, \ldots, S_k), \ldots, D(S_1, \ldots, S_{i_m} \cup \{v\}, \ldots, S_k)$$

отличаются лишь перестановкой индексов множеств частичного разбиения $(S_1, \ldots, S_k)$.

Случай 2. Пусть $m \geqslant 3$. Воспользовавшись пунктом 1 заметим, что суще-

ствует $m-1$ транспозиций $\pi_1 = (i_1, i_2), \ldots, \pi_{m-1} = (i_1, i_m)$, удовлетворяющих условию

$$D(S_1, \ldots, S_{i_1} \cup \{v\}, \ldots, S_k) = D(S_{\pi_1(1)}, \ldots, S_{\pi_1(i_1)}, \ldots, S_{\pi_1(i_2)} \cup \{v\}, \ldots, S_k)$$
$$= \cdots = D(S_{\pi_{m-1}(1)}, \ldots, S_{\pi_{m-1}(i_1)}, \ldots, S_{\pi_{m-1}(i_m)} \cup \{v\}, \ldots, S_k).$$

Утверждение 3.2 доказано. □

Следствие 3.1. *При каждом вызове процедуры* ***Branch*** *можно рассматривать не более чем k подсемейств семейства $D(S_1, \ldots, S_k)$, что значительно сокращает время работы алгоритма.*

Обновление рекорда

Булева функция **updateRecord** проверяет возможность обновить значение текущего рекорда. Так, для задач **GC** и $\mathbf{GC_{\leqslant k}}$ она вычисляет значение логического выражения $b < record$, а для задач $\mathbf{GC_k}$, $\mathbf{SGC_k}$ и $\mathbf{SSGC_k}$ она вычисляет $b < record \& S_1 \neq \emptyset \& \ldots \& S_k \neq \emptyset$. Другими словами, для задач $\mathbf{GC_k}$, $\mathbf{SGC_k}$ и $\mathbf{SSGC_k}$ важно, чтобы построенное разбиение содержало ровно k непустых множеств.

Вычисление оценки

Вычисление нижней оценки значения целевой функции (3.1) для семейства $D(S_1, \ldots, S_k)$ происходит в функции **Bound**. Для построения этой оценки докажем следующее утверждение.

Утверждение 3.3. *Для произвольного графа $G = (V, E), |V| = n$, и произвольного разбиения $(V_1, \ldots, V_k)$ множества вершин V целевую функцию $f_k(V_1, \ldots, V_k)$ можно представить в виде*

$$f_k(V_1, \ldots, V_k) = \frac{n(n-1)}{2} - |E| + 2\sum_{i=1}^{k}\sum_{j=i+1}^{k} c_{ij} - \sum_{i=1}^{k}\sum_{j=i+1}^{k} |V_i||V_j|, \qquad (3.2)$$

где $c_{ij} = c(V_i, V_j) = |uw \in E : u \in V_i, w \in V_j|$ – величина разреза (V_i, V_j).

Доказательство. Нетрудно заметить, что

$$\frac{n(n-1)}{2} = \sum_{i=1}^{k} \frac{|V_i|(|V_i|-1)}{2} + \sum_{i=1}^{k} \sum_{j=i+1}^{k} |V_i||V_j|,$$

$$|E| = \sum_{i=1}^{k} |E_i| + \sum_{i=1}^{k} \sum_{j=i+1}^{k} c_{ij},$$

где E_i – множество ребер подграфа графа G, порожденного множеством вершин V_i. Подставив полученные выражения в формулу (3.2), получаем

$$f_k(V_1, \ldots, V_k) = (\sum_{i=1}^{k} \frac{|V_i|(|V_i|-1)}{2} - |E_i|) + \sum_{i=1}^{k} \sum_{j=i+1}^{k} c_{ij} = \rho(G, M),$$

где $M = M(V_1, \ldots, V_k)$ – кластерный граф, порожденный множествами вершин $V_1, \ldots, V_k$.

Утверждение 3.3 доказано. □

Простейшей оценкой для семейства $D(S_1, \ldots, S_k)$ может служить величина

$$\Delta_1 = |uw \in E : u \in S_i, w \in S_j, i, j \in \{1, \ldots, k\}, i < j|$$
$$+|uw \notin E : u, w \in S_i, i \in \{1, \ldots, k\}|,$$

которая совпадает со значением целевой функции (3.1) на подграфе, порожденном множеством вершин $S_1 \cup \ldots \cup S_k$. Эту оценку можно усилить. Для этого рассмотрим произвольную вершину $w \in V \setminus (S_1 \cup \ldots \cup S_k)$ и вычислим для нее следующие величины

$$b_i(w) = |N_G(w) \cap (S_1 \cup \ldots \cup S_{i-1} \cup S_{i+1} \cup \ldots \cup S_k)| + |\overline{N}_G(w) \cap S_i|.$$

Куда бы ни попала вершина w в конечном разбиении $(V_1, \ldots, V_k) \in D(S_1, \ldots, S_k)$, вносимый ею штраф будет не меньше, чем $\min_{i \in \{1,\ldots,k\}} b_i(w)$. Таким образом, сложив вместе величины $\min_{i \in \{1,\ldots,k\}} b_i(w)$ для всех вершин $w \in V \setminus (S_1 \cup \ldots \cup S_k)$, мы получаем следующую оценку:

$$\Delta_2 = \Delta_1 + \sum_{w \in V \setminus (S_1 \cup \ldots \cup S_k)} \min_{i \in \{1,\ldots,k\}} b_i(w).$$

Эту оценку также можно усилить. Для этого рассмотрим подграф $G(U) = (U, E(U))$ графа G, порожденный множеством вершин $U = V \setminus (S_1 \cup \ldots \cup S_k)$. Оценим снизу значение целевой функции f_k на подграфе $G(U)$, воспользовавшись формулой (3.2). Так как максимум суммы произведений $|U_i||U_j|$ достигается при значениях $|U_1| = \cdots = |U_k| = \frac{|U|}{k}$ и при $2\sum_{i=1}^{k}\sum_{j=i+1}^{k} c_{ij} \leqslant 0$, то справедливо следующее неравенство

$$f_k(U_1, \ldots, U_k) \geqslant \frac{|U|(|U|-k)}{2} - |E(U)|.$$

Очевидно, что если выражение справа больше 0, то оценку Δ_2 можно усилить:

$$\Delta_3 = \Delta_2 + \max\{0, \frac{|U|(|U|-k)}{2} - |E(U)|\}.$$

Выбор вершины для ветвления

Выбирая вершину на шаге 1 процедуры **Branch**, можно управлять процессом ветвления, стараясь как можно быстрее отсечь неперспективные множества. Для этого, имея частичное разбиение $(S_1, \ldots, S_k)$, нужно постараться выбрать вершину v так, чтобы как можно больше семейств $D(S_1 \cup \{v\}, \ldots, S_k)$, $\ldots, D(S_1, \ldots, S_k \cup \{v\})$ имели большую оценку. Так, для алгоритма **BBM** можно реализовать следующее правило – выбирается вершина v, максимизирующая значение выражения

$$\min_{i \in \{1,\ldots,k\}} b_i(v) + |N_G(V) \cap (V \setminus (S_1 \cup \ldots \cup S_k))|.$$

3.1.3 Модели целочисленного линейного программирования

Большинство пакетов прикладных программ, направленных на решение задач оптимизации (CPLEX, GAMS и т.д.), требуют постановки решаемой задачи в виде задачи математического программирования. Данный параграф посвящен построению моделей целочисленного линейного программирования для некоторых из рассмотренных вариантов задач кластеризации вершин графа.

В 2005 г. Чарикар, Гурусвами и Вирт [29] предложили модель булева программирования для задачи **GC**. Рассмотрим произвольный граф $G = (V, E)$,

$V = \{1, \ldots, n\}$. Разбиение вершин на кластеры можно представить с помощью бинарных переменных x_{ij}, заданных для каждой пары вершин. Если вершины i и j находятся в одном множестве разбиения, то $x_{ij} = 0$, иначе $x_{ij} = 1$ при $i \neq j$. По умолчанию будем считать, что $x_{ii} = 0$. Легко видеть, что если $x_{ij} = 0$ и $x_{jr} = 0$, то $x_{ir} = 0$. Следовательно, неравенство $x_{ir} \leqslant x_{ij} + x_{jr}$ выполняется для каждой упорядоченной тройки вершин i, j и r. Это неравенство гарантирует, что любой кластерный граф, являющийся допустимым решением задачи **GC**, не содержит в качестве порожденного подграфа простую цепь P_2 (рис. 1).

Рис. 3.1: Попарно неизоморфные графы с 3 вершинами. Для графа (в) не выполняется неравенство $x_{12} \leqslant x_{13} + x_{23}$.

Благодаря этому свойству получается следующая модель булева программирования для задачи **GC**:

$$\sum_{ij \in E} x_{ij} + \sum_{ij \notin E} (1 - x_{ij}) \to \min \tag{3.3}$$

$$x_{ir} \leqslant x_{ij} + x_{jr}, \qquad i, j, r \in \{1, \ldots, n\} \tag{3.4}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j \in \{1, \ldots, n\} \tag{3.5}$$

Эту модель можно использовать для построения моделей булева программирования задач $\mathbf{GC_2}$, $\mathbf{GC_{\leqslant k}}$, $\mathbf{SGC_k}$ и $\mathbf{SSGC_k}$.

Модель булева программирования для задачи $\mathbf{GC_{\leqslant k}}$

Чтобы построить модель булева программирования для задачи $\mathbf{GC_{\leqslant k}}$, необходимо ограничить сверху мощность разбиения множества V. Заметим, что кластерный граф, являющийся допустимым решением задачи $\mathbf{GC_{\leqslant k}}$, не может содержать в качестве порожденного подграфа пустой граф O_{k+1}. Это означает, что для каждого упорядоченного набора $i_1, \ldots, i_k$, состоящего из k различных вершин графа G, будет выполняться следующее неравенство

(пример для $k = 3$ на рис 3.2):

$$x_{i_1 i_2} + \ldots x_{i_{k-1} i_k} \leqslant \binom{k+1}{2} - 1 = \frac{(k+2)(k-1)}{2}.$$

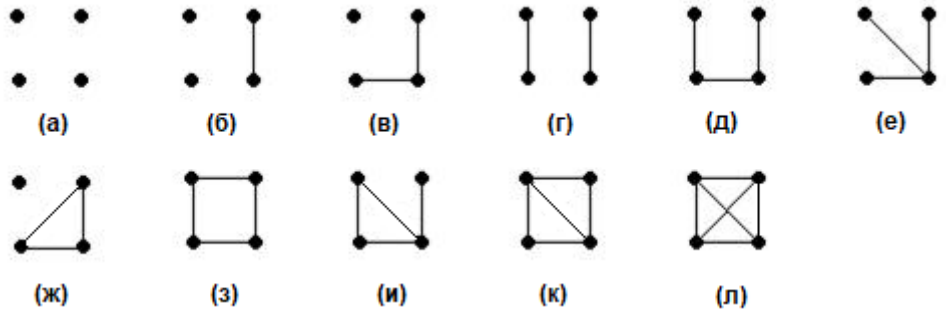

Рис. 3.2: Попарно неизоморфные графы с 4 вершинами. При $k = 3$ граф (а) запрещен, поскольку является пустым графом O_4. Графы (в), (д), (е), (з), (и), (к) запрещены, поскольку содержат в качестве порожденного подграфа простую цепь P_2.

Добавив это ограничение к модели (3.3) – (3.5), мы получаем модель булева программирования для задачи $\mathbf{GC}_{\leqslant \mathbf{k}}$.

$$\sum_{ij \in E} x_{ij} + \sum_{ij \notin E} (1 - x_{ij}) \to \min \tag{3.6}$$

$$x_{ir} \leqslant x_{ij} + x_{jr}, \qquad i, j, r \in \{1, \ldots, n\} \tag{3.7}$$

$$x_{i_1 i_2} + \ldots x_{i_{k-1} i_k} \leqslant \frac{(k+2)(k-1)}{2}, \qquad i_1, \ldots, i_k \in \{1, \ldots, n\} \tag{3.8}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j \in \{1, \ldots, n\} \tag{3.9}$$

Модель булева программирования для задачи $\mathbf{GC_2}$

Чтобы построить модель булева программирования для задачи $\mathbf{GC_k}$ необходимо ограничить снизу мощность разбиения множества V, что является непростой задачей. Однако, в случае $k = 2$ такое ограничение строится легко. Ограничение (3.8) в случае $k = 2$ превращается в неравенство $x_{ij} + x_{ir} + x_{jr} \leqslant 2$, которое будет выполнено и в случае полного графа K_n.

Заметим, что в K_n для каждой вершины $i \in \{1, \ldots, n\}$выполняется равенство $\sum_{j=1}^{n} x_{ij} = 0$. Чтобы запретить K_n как допустимое решение, достаточно ввести в модель (3.6) – (3.9) дополнительное ограничение $\sum_{j=1}^{n} x_{ij} \geqslant 1$ для

некоторой вершины $i \in \{1, \ldots, n\}$. Сделаем это для вершины $i = 1$.

$$\sum_{ij \in E} x_{ij} + \sum_{ij \notin E} (1 - x_{ij}) \to \min \tag{3.10}$$

$$x_{ir} \leqslant x_{ij} + x_{jr}, \qquad i, j, r \in \{1, \ldots, n\} \tag{3.11}$$

$$x_{ij} + x_{ir} + x_{jr} \leqslant 2, \qquad i, j, r \in \{1, \ldots, n\} \tag{3.12}$$

$$\sum_{j=1}^{n} x_{1j} \geqslant 1 \tag{3.13}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j \in \{1, \ldots, n\} \tag{3.14}$$

Модель булева программирования для задач $\mathbf{SGC_k}$ и $\mathbf{SSGC_k}$

В отличие от задачи $\mathbf{GC_k}$, в задачах $\mathbf{SGC_k}$ и $\mathbf{SSGC_k}$ ограничить снизу мощность разбиения множества V достаточно просто. Так, для задачи $\mathbf{SGC_k}$ нужно потребовать, чтобы $x_{zz'} = 1$ для всех пар вершин $z, z' \in Z$.

$$\sum_{ij \in E} x_{ij} + \sum_{ij \notin E} (1 - x_{ij}) \to \min \tag{3.15}$$

$$x_{ir} \leqslant x_{ij} + x_{jr}, \qquad i, j, r \in \{1, \ldots, n\} \tag{3.16}$$

$$x_{i_1 i_2} + \ldots x_{i_{k-1} i_k} \leqslant \frac{(k+2)(k-1)}{2}, \qquad i_1, \ldots, i_k \in \{1, \ldots, n\} \tag{3.17}$$

$$x_{zz'} = 1, \qquad z, z' \in Z \tag{3.18}$$

$$x_{ij} \in \{0, 1\}, \qquad i, j \in \{1, \ldots, n\} \tag{3.19}$$

Для задачи $\mathbf{SSGC_k}$ нужно потребовать, чтобы $x_{zz'} = 1$ если z и z' принадлежат разным множествам семейства $\mathcal{Z}$ и $x_{zz'} = 0$ если z и z' принадлежат одному множеству семейства $\mathcal{Z}$.

$$\sum_{ij \in E} x_{ij} + \sum_{ij \notin E} (1 - x_{ij}) \to \min \tag{3.20}$$

$$x_{ir} \leqslant x_{ij} + x_{jr}, \qquad i,j,r \in \{1,\ldots,n\} \quad (3.21)$$

$$x_{i_1 i_2} + \ldots x_{i_{k-1} i_k} \leqslant \frac{(k+2)(k-1)}{2}, \qquad i_1,\ldots,i_k \in \{1,\ldots,n\} \quad (3.22)$$

$$x_{zz'} = 0, \qquad z,z' \in Z_i, i \in \{1,\ldots,n\} \quad (3.23)$$

$$x_{zz'} = 1, \qquad z \in Z_i, z' \in Z_j, i,j \in \{1,\ldots,n\} \quad (3.24)$$

$$x_{ij} \in \{0,1\}, \qquad i,j \in \{1,\ldots,n\} \quad (3.25)$$

3.2 Экспериментальное исследование приближенных и точных алгоритмов для задач кластеризации вершин графа

Результаты глав 1 и 2, а также параграфа 3.1 дают основания для анализа эффективности приближенных и точных алгоритмов решения задач кластеризации вершин графа. Одним из распространенных подходов в исследовании алгоритмов является вычислительный эксперимент. В этом параграфе предлагается описание экспериментального исследования описанных выше приближенных и точных алгоритмов, а также новых эвристических алгоритмов для задач $\mathbf{GC_{\leqslant 2}}$, $\mathbf{GC_2}$, $\mathbf{SSGC_2}$ и $\mathbf{GC_{\leqslant 3}}$. Цели эксперимента состояли в том, чтобы на основе статистических данных оценить

1. время работы приближенных и точных алгоритмов,
2. точность приближенных алгоритмов,
3. целесообразность многократного использования процедуры локального поиска.

3.2.1 Описание вычислительного эксперимента

Прежде чем говорить о каких-либо результатах экспериментального исследования, необходимо описать модель, в рамках которой это исследование проводилось. Цель нашего эксперимента будет заключаться в том, чтобы сравнить различные приближенные и точные алгоритмы по нескольким критериям (трудоемкость, точность) между собой и выявить лучший из них для каждой из задач.

Схему проведения вычислительного эксперимента, направленного на достижение поставленной цели, можно описать так.

1. Вводится вероятностное распределение на множестве входов исследуемой задачи, то есть задается вероятностное пространство на множестве графов.

2. В соответствии с вероятностным распределением проводится случайный выбор N графов, на которых исследуемыми алгоритмами решается задача.

3. Для каждого полученного решения вычисляется время работы и точность, которые является случайной величиной.

4. На основе статистических вычисляются оценки математического ожидания времени работы и точности исследуемых алгоритмов.

Итак, первым делом введем на множестве всех графов вероятностное распределение. Для этого зафиксируем параметр $p \in (0, 1)$. Случайный n-вершинный граф $G = (V, E)$ будем получать с использованием следующей процедуры. Для каждой пары вершин (u, v) проводится независимый случайный эксперимент, исходами которого будут наличие ребра uv с вероятностью p и отсутствие ребра с вероятностью $1 - p$. Таким образом, граф G можно рассматривать как случайный вектор, каждая координата которого соотвествует паре вершин графа G. В литературе [19, 21, 27] семейство n-вершинных графов с введенным таким образом распределением обозначается $G(n, p)$ и используется как при теоретическом изучении графов, так и в экспериментальных исследованиях (модель Эрдёша-Реньи).

Параметр p в используемой вероятностной модели представляет собой математическое ожидание плотности случайного графа $G = (V, E)$, которая определяется как $\frac{2|E|}{n(n-1)}$. В проведенных экспериментах использовались значения p из множества $\{0.33, 0.5, 0.67\}$.

Рассмотрим теперь параметр n семейства $G(n, p)$. Главным ограничением на количество вершин в тестовых задачах была сложность отыскания оптимального значения целевой функции, необходимого для вычисления точности каждого изучаемого алгоритма. По этой причине экспериментальное

исследование можно разделить на две части: предварительный эксперимент на графах малой размерности и основной эксперимент на графах большей размерности.

Предварительный эксперимент проводился на n-вершинных графах, значение n зависело от исследуемой задачи. В рамках эксперимента для каждой пары значений параметров n и p было решено по 100 задач. Оптимальные решения были найдены алгоритмом полного перебора **BF**, алгоритмом ветвей и границ **BBM**, а также двумя решателями **Gurobi** и **CBC**, опирающимися на модели целочисленного линейного программирования.

Для оценки качества полученного приближенного решения использовалась величина (*точность*), определяемая равенством

$$\delta_A = \frac{f(A)}{f(OPT)},$$

где $f(A)$ – значение целевой фунции, полученное некоторым приближенным алгоритмом A, а $f(OPT)$ – значение целевой функции, полученное точным алгоритмом OPT. Поскольку δ есть функция от случайного графа, то она сама является случайной величиной. Задача предварительного эксперимента состояла в том, чтобы оценить время работы точных алгоритмов, а также найти оценку математического ожидания случайной величины δ. Для ее вычисления использовалась стандартная формула для среднего значения:

$$\overline{\delta}_A = \frac{\sum_{i=1}^{N} \delta_i}{N},$$

где δ_i – точность алгоритма в i-ом эксперименте, $N = 100$.

По итогам предварительного эксперимента удалось получить представление о характере поведения точности алгоритмов и сформулировать некоторые гипотезы.

Основной эксперимент проводился на графах, размерность которых также зависела от исследуемой задачи. Для каждой пары n и p было сгенерировано и решено по 100 задач. Цель основного эксперимента состояла в том, чтобы подтвердить или опровергнуть гипотезы, выдвинутые по результатам предварительного эксперимента.

Все перечисленные точные и приближенные алгоритмы были реализованы в виде программ на языке программирования C++ в среде JetBrains Clion 2020.1. Вычисления проводились на персональном компьютере с процессором Intel Core i7-9700, тактовой частотой 3.0 ГГц и объемом оперативной памяти, равным 32 Гбайта. Решатели **Gurobi** и **CBC** использовались в программе, написанной на языке программирования Python в среде IntelliJ IDEA 2019.3.

3.2.2 Экспериментальное исследование алгоритмов для задач $\mathbf{GC_{\leqslant 2}}$, $\mathbf{GC_2}$, $\mathbf{SGC_2}$ и $\mathbf{GC_{\leqslant 3}}$

Экспериментальное исследование алгоритмов для задачи $\mathbf{GC_{\leqslant 2}}$

В этом эксперименте, помимо алгоритмов **BBC** (Bansal-Blum-Chawla) [23] и **CSW** (Coleman-Saunderson-Wirth) [32] был предложен эвристический алгоритм $\mathbf{N1LS_{\leqslant 2}}$ (Neighborhood with one Local Search for $\mathbf{GC_{\leqslant 2}}$). Ключевое изменение этого алгоритма в сравнении с алгоритмом **CSW** – процедура локального поиска $\mathbf{LS_{\leqslant 2}}$ применяется лишь к лучшему допустимому решению, полученному алгоритмом **BBC**, что значительно сокращает время его работы (поскольку трудоемкость этого алгоритма – $O(n^3)$). Также его гарантированная оценка точности не хуже, чем гарантированная оценка точности алгоритма **BBC**.

Предварительный эксперимент проводился на графах размерности от 15 до 50 вершин из пространства $G(n, p)$. Для таких графов алгоритмом ветвей и границ **BBM**, алгоритмом полного перебора **BF** и с помощью решателей **Gurobi** и **CBC**, использующих модель целочисленного линейного программирования (3.6) – (3.7), удалось найти точные решения. Это позволило сделать вывод о времени работы точных алгоритмов, а также сделать некоторые предположения об изменении точности алгоритмов в зависимости от параметров n и p.

Время работы точных алгоритмов (таблицы 3.1 – 3.3 приложения) распределилось следующим образом. Худшие результаты принадлежат решателю **CBC**. Его время работы при $n = 25$ стало выше 1000 секунд, что хуже показателей алгоритма полного перебора **BF**, время работы которого приближается к 1000 сек при $n = 30$. Решатель **Gurobi** справился с задачей в

случае $n = 35$, хотя его время работы также было больше 1000 секунд. Наилучшие результаты принадлежат алгоритму **BBM**, который решил задачу для графов с 50 вершинами. При этом стоит отметить, что лишь в случае $p = 0.5$ время его работы было равно 1781.96 секунд. В случаях $p = 0.33$ и $p = 0.67$ среднее вермя его работы составило, соотвественно, 262.72 и 110.03 секунды.

Поведение точности приближенных алгоритмов на графах малой размерности значительно отличалось в случае $p \in \{0.33, 0.5\}$ и в случае $p = 0.67$.

В случае $p \in \{0.33, 0.5\}$ средняя ошибка алгоритмов **BBC** и $\mathbf{N1LS_{\leqslant 2}}$ имела тенденцию к убыванию (рис. 3.3). Максимальное значение средней ошибки алгоритмов **BBC** и $\mathbf{N1LS_{\leqslant 2}}$ составило порядка 16% и 4% в случае $p = 0.33, n = 20$. Средняя ошибка алгоритма **CSW** была около 0.

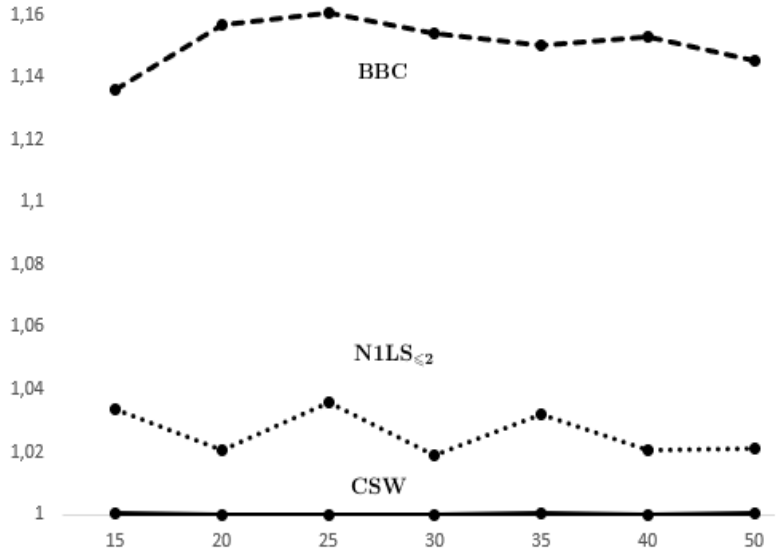

Рис. 3.3: Средняя точность алгоритмов **BBC**, **CSW** и $\mathbf{N1LS_{\leqslant 2}}$ на графах малой размерности при $p = 0.33$.

В случае $p = 0.67$ поведение алгоритмов **BBC** и $\mathbf{N1LS_{\leqslant 2}}$ сильно меняется (рис. 3.4). Так, средняя ошибка алгоритма $\mathbf{N1LS_{\leqslant 2}}$ стремится к 0 с ростом n. Средняя ошибка алгоритма **BBC** наоборот, увеличивается с ростом n и составляет порядка 25% в случае $n = 50$. Средняя ошибка алгоритма **CSW** также остается около 0.

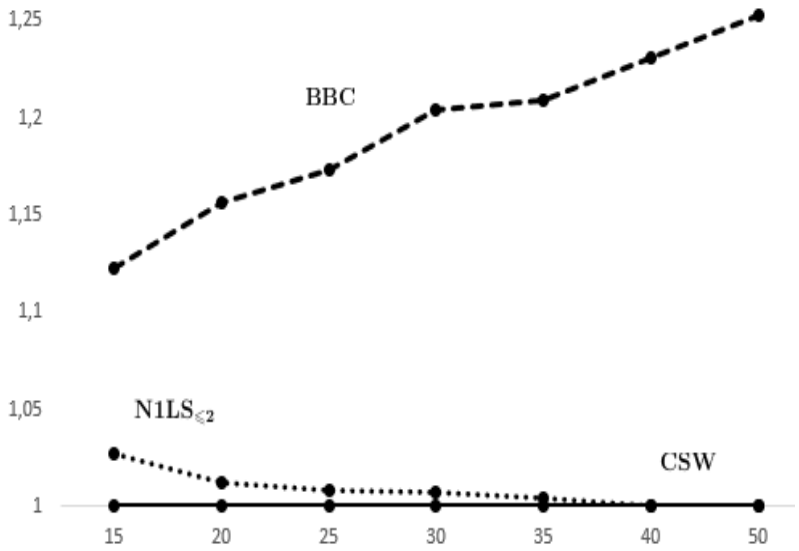

Рис. 3.4: Средняя точность алгоритмов **BBC**, **CSW** и $\mathbf{N1LS_{\leqslant 2}}$ на графах малой размерности при $p = 0.67$.

По результатам предварительного эксперимента появляется надежда на то, что точность алгоритма $\mathbf{N1LS_{\leqslant 2}}$ в среднем не хуже точности алгоритма **CSW** при росте n. При этом точность алгоритма **BBC** вряд ли приблизится к точности алгоритма **CSW** в случае $p = 0.67$.

Очевидно, что точность алгоритма **CSW** всегда не меньше, чем точность алгоритмов **BBC** и $\mathbf{N1LS_{\leqslant 2}}$, поэтому в качестве объекта дальнейшего исследования были выбраны две случайные величины:

$$d_{\mathbf{BBC}}(n) = \frac{\delta_{\mathbf{BBC}}(n)}{\delta_{\mathbf{CSW}}(n)},\ d_{\mathbf{N1LS_{\leqslant 2}}}(n) = \frac{\delta_{\mathbf{N1LS_{\leqslant 2}}}(n)}{\delta_{\mathbf{CSW}}(n)}.$$

Выбор таких величин для исследования можно объяснить тем, что, во-первых, для их вычисления не требуется знание оптимального решения, во-вторых, они могут не только ответить на вопрос, какой из алгоритмов имеет преимущество над другими, но и позволяют оценить во сколько раз решение, найденное одним из алгоритмов, лучше решений, полученных другими двумя алгоритмами.

По результатам предварительного эксперимента были выдвинуты основные гипотези, проверка которой методами математической статистики осуществлена далее.

Гипотеза 3.1. *При достаточно больших n математическое ожидание случайной величины* $d_{\mathbf{BBC}}(n)$ *не меньше 1.*

Гипотеза 3.2. *При достаточно больших n математическое ожидание случайной величины* $d_{\mathbf{N1LS_{\leqslant 2}}}(n)$ *стремится к 1.*

Для проверки гипотез был проведен эксперимент на графах $G(n,p)$ большей размерности (при n от 100 до 6000). Проверка осуществлялась по широко распространенной схеме: на основе выборочных данных были получены оценки $\overline{d}_{\mathbf{BBC}}(n)$ и $\overline{d}_{\mathbf{N1LS_{\leqslant 2}}}(n)$ математического ожидания для каждой из исследуемых случайных величин, после чего при уровне значимости $\alpha = 0.05$ расчитаны границы доверительных интервалов.

Для расчета границ доверительного интервала использовался квантиль нормального распределения. Статистическое обоснование этому было получено путем проверки критерия согласия Колмогорова-Смирнова эмпирической функции распределения $F_{\mathbf{BBC}}(x)$ с соответствующей ей функцией нормального распределения, имеющей параметры $\overline{d}_{\mathbf{BBC}}(n)$ и $\overline{\sigma}_{\mathbf{BBC}}$. Тоже самое было сделано для случайной величины $d_{\mathbf{N1LS_{\leqslant 2}}}(n)$.

В таблицах 3.7 и 3.11 приложения приведены значения статистик Колмогорова-Смирнова при различных n и p. Максимум из этих значений достигается при $n = 6000$ и $p = 0.33$ и равно 1.32. Это меньше, чем 1.36 – квантиль распределения Колмогорова-Смирнова при уровне значимости 0.05, что позволяет принять гипотезу о нормальности распределений случайных величин $d_{\mathbf{BBC}}(n)$ и $d_{\mathbf{N1LS_{\leqslant 2}}}(n)$.

Как и в случае предварительного эксперимента на графах малой размерности, поведение алгоритмов при $p \in \{0.33, 0.5\}$ и при $p = 0.67$ отличалось.

При $p \in \{0.33, 0.5\}$ средняя относительная ошибка алгоритмов $d_{\mathbf{BBC}}(n)$ и $d_{\mathbf{N1LS_{\leqslant 2}}}(n)$ падала (рис 3.5). Величина $d_{\mathbf{N1LS_{\leqslant 2}}}(n)$ вообще находится в окрестности единицы.

При $p = 0.67$ средняя относительная ошибка алгоритма **BBC** росла. Алгоритм $\mathbf{N1LS_{\leqslant 2}}$ дает такие же решение, что и алгоритм **CSW**, т.е. его ошибка вообще равна 1 (рис 3.6).

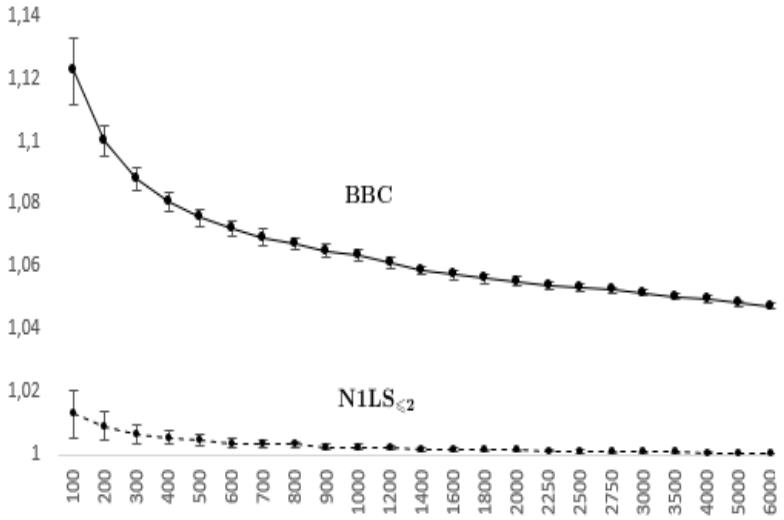

Рис. 3.5: Средние значения случайных величин $d_{\mathbf{BBC}}(n)$ и $d_{\mathbf{N1LS_{\leqslant 2}}}(n)$ при $p = 0.33$.

По результатам вычислительного эксперимента можно говорить о том, что достаточных оснований для отклонения гипотез 3.1 и 3.2 нет. Точность алгоритма $\mathbf{N1LS_{\leqslant 2}}$ при $p \in \{0.33, 0.5\}$ почти вплотную подходит к 1 с ростом n, а ширина доверительного интервала сужается настолько, что интервалы фактически невозможно увидеть на рисунке. При $p \in \{0.67\}$ решения алгоритма $\mathbf{N1LS_{\leqslant 2}}$ полностью совпадают с решениями алгоритма **CSW**. Это говорит о том, что на графах большой плотности алгоритм $\mathbf{N1LS_{\leqslant 2}}$ вообще не проигрывает алгоритму **CSW**. Учитывая, что при $n = 6000$ среднее время работы алгоритма $\mathbf{N1LS_{\leqslant 2}}$ составило 149.21 сек, а среднее время работы алгоритма **CSW** составило 580.68 сек, использование алгоритма $\mathbf{N1LS_{\leqslant 2}}$ является наиболее предпочтительным с практической точки зрения.

Говоря отдельно об алгоритме **BBC** можно сказать, что на графах небольшой плотности его относительная средняя ошибка падает с ростом n. Это оставляет надежду на то, что при $n \to \infty$ его средняя ошибка также стремится к средней ошибке алгоритма **CSW**. Однако, из-за значительного роста его относительной средней ошибки в случае $p = 0.67$, данный алгоритм нельзя считать хорошей альтернативой алгоритмам **CSW** и $\mathbf{N1LS_{\leqslant 2}}$. Дополнительным плюсом для алгоритма $\mathbf{N1LS_{\leqslant 2}}$ служит незначительное увеличение его среднего времени работы при однократном использовании процедуры локального поиска относительно алгоритма **BBC** (149.21 сек против 149.58 сек).

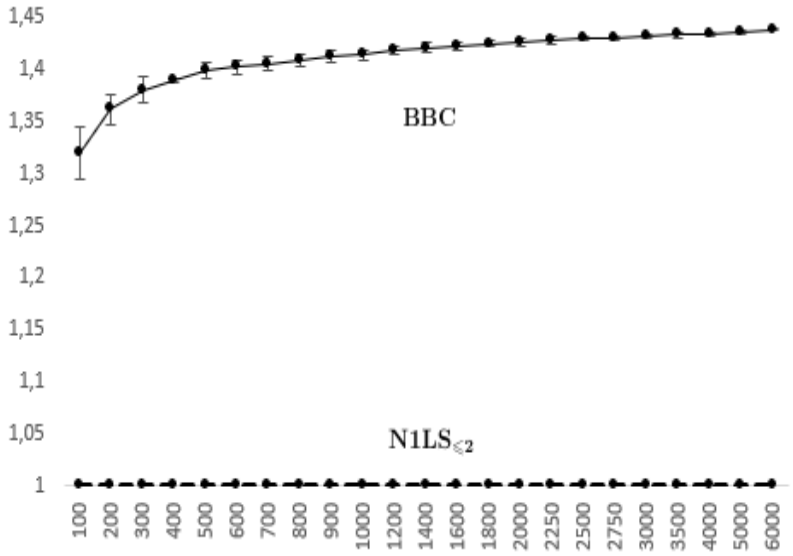

Рис. 3.6: Средние значения случайных величин $d_{\mathbf{BBC}}(n)$ и $d_{\mathbf{N1LS_{\leqslant 2}}}(n)$ при $p = 0.67$.

Полные сведения о результатах эксперимента на графах большой размерности приведены в таблицах 3.1 – 3.11 приложения.

Экспериментальное исследование алгоритмов для задачи $\mathbf{GC_2}$

В этом эксперименте, по аналогии с экспериментом для задачи $\mathbf{GC_{\leqslant 2}}$, наряду с алгоритмами $\mathbf{N_2}$ и $\mathbf{NLS_2}$ был исследован алгоритм $\mathbf{N1LS_2}$ (Neighborhood with one Local Search for $\mathbf{GC_2}$). В отличие от алгоритма $\mathbf{NLS_2}$, этот алгоритм единожды вызывает процедуру локального поиска $\mathbf{LS_2}$, применяя ее лишь к допустимому решению, полученному алгоритмом $\mathbf{N_2}$, тем самым сокращая время работы и сохраняя гарантированную оценку точности алгоритма $\mathbf{N_2}$.

Предварительный эксперимент проводился на графах размерности от 10 до 50 вершин из пространства $G(n, p)$. Для таких графов алгоритмом ветвей и границ **BBM**, алгоритмом полного перебора **BF** и с помощью решателей **Gurobi** и **CBC**, использующих модель целочисленного линейного программирования (3.10) – (3.14), удалось найти точные решения. Это позволило сделать вывод о времени работы точных алгоритмов, а также сделать некоторые предположения об изменении точности алгоритмов в зависимости от параметров n и p.

Время работы точных алгоритмов (таблицы 3.12 – 3.13 приложения) распределилось следующим образом. Худшие результаты опять принадлежат решателю **CBC**. Его время работы при $n = 25$ и $p = 0.5$ стало выше 2000 се-

кунд, что хуже показателей алгоритма полного перебора **BF**, время работы которого приближается к 1000 сек при $n = 30$. Решатель **Gurobi** справился с задачей в случае $n = 35$, хотя его время работы также было больше 2000 секунд. Наилучшие результаты принадлежат алгоритму **BBM**, который решил задачу для графов с 50 вершинами. При этом стоит отметить, что как и в случае эксперимента для задачи $\mathbf{GC}_{\leqslant \mathbf{2}}$, лишь в случае $p = 0.5$ время его работы было равно 2003.14 секунды. В случаях $p = 0.33$ и $p = 0.67$ среднее вермя его работы составило, соотвественно, 259.75 и 112.6 секунды.

Поведение точности приближенных алгоритмов на графах малой размерности было схоже при всех значениях p. Средняя ошибка алгоритмов $\mathbf{N_2}$ имела тенденцию к возрастанию, средняя ошибка алгоритма $\mathbf{N1LS_2}$ убывала, а средняя ошибка алгоритма $\mathbf{NLS_2}$ мало отличалась от 0 (рис. 3.7).

По результатам предварительного эксперимента были сформулированы следующие наблюдения: средняя ошибка $\mathbf{N_2}$ должна и дальше расти с ростом n, в то время как средние ошибки $\mathbf{NLS_2}$ и $\mathbf{N1LS_2}$ сохранят тенденцию к убыванию или не будут меняться.

Для дальнейшего исследования были введены две случайные величины:

$$d_{\mathbf{N_2}}(n) = \frac{\delta_{\mathbf{N_2}}(n)}{\delta_{\mathbf{NLS_2}}(n)},\ d_{\mathbf{N1LS_2}}(n) = \frac{\delta_{\mathbf{N1LS_2}}(n)}{\delta_{\mathbf{NLS_2}}(n)}.$$

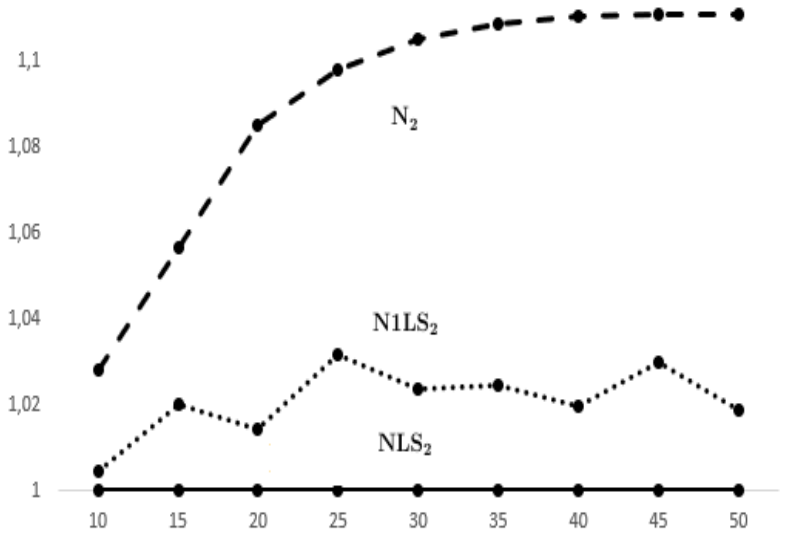

Рис. 3.7: Средняя точность алгоритмов $\mathbf{N_2}$, $\mathbf{NLS_2}$ и $\mathbf{N1LS_2}$ на графах малой размерности при $p = 0.5$.

Были сформулированы следующие гипотезы, проверка которых осуществлена на графах большей размерности.

Гипотеза 3.3. *При достаточно больших n математическое ожидание случайной величины* $d_{\mathbf{N1LS_2}}(n)$ *стремится к 1.*

Гипотеза 3.4. *При достаточно больших n математическое ожидание случайной величины* $d_{\mathbf{N_2}}(n)$ *не меньше 1.*

Для проверки поведения случайных величин был проведен эксперимент на графах $G(n,p)$ большей размерности (при n от 100 до 600). Стоит заметить, что трудоемкость всех приближенных алгоритмов для задачи $\mathbf{GC_2}$ на порядок выше, чем трудоемкость соответствующих им приближенных алгоритмов для задачи $\mathbf{GC_{\leqslant 2}}$. Это не позволяет исследовать задачу на графах большой размерности. На основе выборочных данных были получены оценки $\overline{d}_{\mathbf{N_2}}(n)$ и $\overline{d}_{\mathbf{N1LS_2}}(n)$ математического ожидания для каждой из исследуемых случайных величин, после чего при уровне значимости $\alpha = 0.05$ расчитаны границы доверительных интервалов.

Для расчета границ доверительного интервала использовался квантиль нормального распределения. Статистическое обоснование этому было получено путем проверки критерия согласия Колмогорова-Смирнова эмпирической функции распределения $F_{\mathbf{N_2}}(x)$ с соответствующей ей функцией нормального распределения, имеющей параметры $\overline{d}_{\mathbf{N_2}}(n)$ и $\overline{\sigma}_{\mathbf{N_2}}(n)$. То же самое было сделано для случайной величины $d_{\mathbf{N1LS_2}}(n)$.

В таблицах 3.18 и 3.22 приложения приведены значения статистик Колмогорова–Смирнова при различных n и p. Максимум из этих значений достигается при $n = 100$ и $p = 0.33$ и равен 1.25. Это меньше, чем 1.36 – квантиль распределения Колмогорова-Смирнова при уровне значимости 0.05, что позволяет принять гипотезу о нормальности распределений случайных величин $d_{\mathbf{N_2}}(n)$ и $d_{\mathbf{N1LS_2}}(n)$.

На графах большей размерности поведение алгоритма $\mathbf{N_2}$ стало отличаться от поведения на графах малой размерности и разделилось на случаи $p \in \{0.33, 0.5\}$ и $p = 0.67$. Поведение алгоритма $\mathbf{N1LS_2}$ сохранило ту же тенденцию.

При $p \in \{0.33, 0.5\}$ относительная средняя ошибка алгоритма $d_{\mathbf{N_2}}(n)$ начала уменьшаться, как и относительная средняя ошибка $d_{\mathbf{N1LS_2}}(n)$ (рис 3.8).

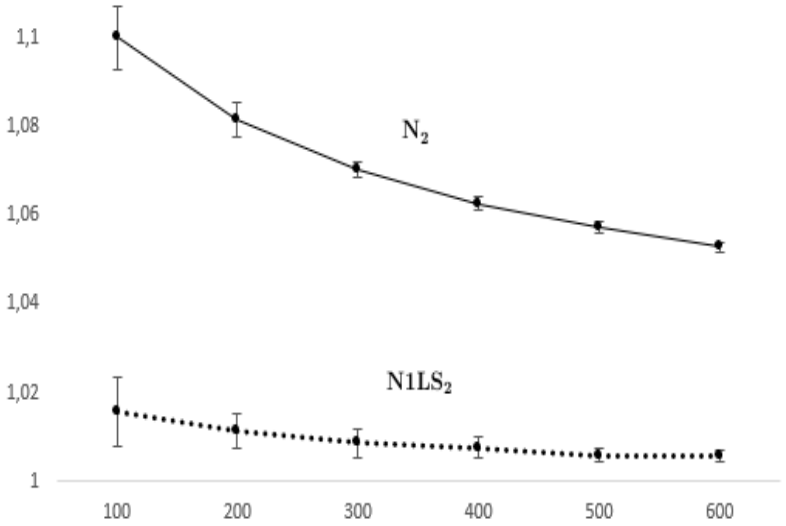

Рис. 3.8: Средние значения случайных величин $d_{\mathbf{N_2}}(n)$ и $d_{\mathbf{N1LS_2}}(n)$ при $p = 0.5$.

При $p = 0.67$ относительная ошибка алгоритма $\mathbf{N_2}$ растет, как и на графах малой размерности. Ошибка алгоритма $\mathbf{N1LS_2}$ находится в окрестности 1 (рис. 3.9).

По результатам вычислительного эксперимента можно говорить о том, что достаточных оснований для отклонения гипотез 3.3 и 3.4 нет. Однако стоит отметить, что при $p \in \{0.33, 0.5\}$ средняя ошибка алгоритма $\mathbf{N_2}$ падает, хотя и растет при $p = 0.67$. В любом случае, использование алгоритма $\mathbf{N1LS_2}$ является наиболее предпочтительным, поскольку его средняя ошибка стремится к средней ошибке алгоритма $\mathbf{NLS_2}$, при этом время его работы фактически совпадает со временем работы алгоритма $\mathbf{N_2}$ и значительно меньше времени работы алгоритма $\mathbf{NLS_2}$.

Полные сведения о результатах эксперимента на графах большей размерности приведены в таблицах 3.12 – 3.22 приложения.

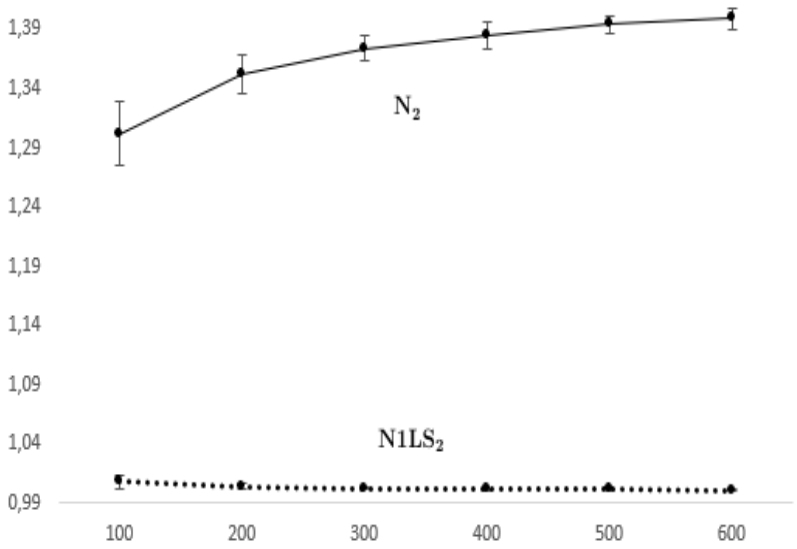

Рис. 3.9: Средние значения случайных величин $d_{\mathbf{N_2}}(n)$ и $d_{\mathbf{N1LS_2}}(n)$ при $p = 0.67$.

Экспериментальное исследование алгоритмов для задачи $\mathbf{SGC_2}$

Для исследования задач кластеризации вершин графа с частичным обучением была выбрана задача $\mathbf{SGC_2}$. Вместе с алгоритмами $\mathbf{NS_2}$ и $\mathbf{NSLS_2}$ был исследован эвристический алгоритм $\mathbf{PNSLS_2}$ (Pre-clustered neighborhood semi-supervised with Local Search for $\mathbf{SGC_2}$). Суть этого алгоритма заключается в следующем: для вершин z_1 и z_2 строится кластерный граф, как на шаге 1 алгоритма $\mathbf{NS_2}$. Далее к каждому построенному кластерному графу применяется процедура локального поиска $\mathbf{LS_2}$. Таким образом, количество построенных кластерных графов не зависит от количества вершин в графе $G = (V, E)$. Алгоритм локального поиска также применяется лишь к ограниченному числу кластерных графов, т.е. трудоемкость алгоритма $\mathbf{PNSLS_2}$ ограничена лишь трудоемкость процедуры локального поиска (другими словами, трудоемкость алгоритма $\mathbf{PNSLS_2}$ – $O(n^3)$). Этот эвристический алгоритм не имеет доказанной априорной гарантированной оценки точности.

Как и ранее, был проведен предварительный эксперимент на графах размерности от 15 до 50 вершин из пространства $G(n, p)$. Для таких графов алгоритмом ветвей и границ **BBM**, алгоритмом полного перебора **BF** и с помощью решателей **Gurobi** и **CBC**, использующих модель целочисленного линейного программирования (3.15) – (3.19), удалось найти точные решения. Это позволило сделать вывод о времени работы точных алгоритмов, а также

сделать предположения об изменении точности алгоритмов в зависимости от параметров n и p.

Время работы точных алгоритмов (таблицы 3.23 – 3.25 приложения) распределилось следующим образом. Как и в двух предыдущих случаях худшие результаты принадлежат решателю **CBC**. Его время работы при $n = 25$ и $p = 0.5$ превысили 700 сек, что хуже показателей алгоритма полного перебора **BF**, время работы которого приближается к 1000 сек при $n = 30$. Решатель **Gurobi** справился с задачей в случае $n = 35$, при этом он намного лучше справился с задачей $\mathbf{SGC_2}$, чем с $\mathbf{GC_{\leqslant 2}}$ и $\mathbf{GC_2}$. Наилучшие результаты вновь принадлежат алгоритму **BBM**, который решил задачу для графов с 50 вершинами. При $p = 0.5$ его время работы составило 1041.37 сек, что значительно лучше его показателей для задач $\mathbf{GC_{\leqslant 2}}$ и $\mathbf{GC_2}$.

Поведение точности приближенных алгоритмов на графах малой размерности было схоже при всех значениях p. Средняя ошибка алгоритмов $\mathbf{NS_2}$ имела тенденцию к неубыванию, средняя ошибка алгоритма $\mathbf{PNSLS_2}$ не возрастала, а погрешность алгоритма $\mathbf{NSLS_2}$ мало отличалась от 0 (рис. 3.10).

Для дальнейшего исследования были введены две случайные величины:

$$d_{\mathbf{NS_2}}(n) = \frac{\delta_{\mathbf{NS_2}}(n)}{\delta_{\mathbf{NSLS_2}}(n)},\ d_{\mathbf{PNSLS_2}}(n) = \frac{\delta_{\mathbf{PNSLS_2}}(n)}{\delta_{\mathbf{NSLS_2}}(n)}.$$

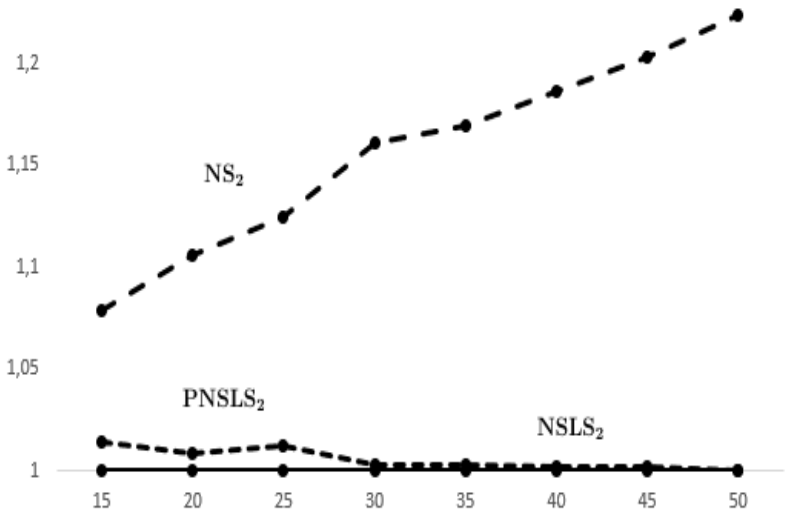

Рис. 3.10: Средняя точность алгоритмов $\mathbf{NS_2}$, $\mathbf{NSLS_2}$ и $\mathbf{PNSLS_2}$ на графах малой размерности при $p = 0.67$.

Были сформулированы следующие гипотезы, проверка которых осуществлена на графах большей размерности.

Гипотеза 3.5. *При достаточно больших n математическое ожидание случайной величины* $d_{\mathbf{NS_2}}(n)$ *не меньше 1.*

Гипотеза 3.6. *При достаточно больших n математическое ожидание случайной величины* $d_{\mathbf{PNSLS_2}}(n)$ *стремится к 1.*

Для проверки гипотез был проведен эксперимент на графах $G(n,p)$ большей размерности (при n от 100 до 3000). На основе выборочных данных были получены оценки $\overline{d}_{\mathbf{NS_2}}(n)$ и $\overline{d}_{\mathbf{PNSLS_2}}(n)$ математического ожидания для каждой из исследуемых случайных величин, после чего при уровне значимости $\alpha = 0.05$ расчитаны границы доверительных интервалов.

Для расчета границ доверительного интервала использовался квантиль нормального распределения. Статистическое обоснование этому было получено путем проверки критерия согласия Колмогорова-Смирнова эмпирической функции распределения $F_{\mathbf{NS_2}}(x)$ с соответствующей ей функцией нормального распределения, имеющей параметры $\overline{d}_{\mathbf{NS_2}}(n)$ и $\overline{\sigma}_{\mathbf{NS_2}}(n)$. Тоже самое было сделано для случайной величины $d_{\mathbf{PNSLS_2}}(n)$.

В таблицах 3.29 и 3.33 приложения приведены значения статистик Колмогорова–Смирнова при различных n и p. Максимум из этих значений достигается при $n = 800$ и $p = 0.67$ и равен 1.19. Это меньше, чем 1.36 – квантиль распределения Колмогорова-Смирнова при уровне значимости 0.05, что позволяет принять гипотезу о нормальности распределений случайных величин $d_{\mathbf{NS_2}}(n)$ и $d_{\mathbf{PNSLS_2}}(n)$.

На графах большей размерности поведение алгоритма $\mathbf{NS_2}$ начало отличаться от поведения на графах малой размерности и разделилось на два случая $p \in \{0.33, 0.5\}$ и $p = 0.67$. Поведение алгоритма $\mathbf{PNSLS_2}$ сохранило ту же тенденцию.

При $p \in \{0.33, 0.5\}$ относительная средняя ошибка алгоритма $d_{\mathbf{NS_2}}(n)$ начала уменьшаться, как и относительная средняя ошибка $d_{\mathbf{PNSLS_2}}(n)$ (рис 3.11).

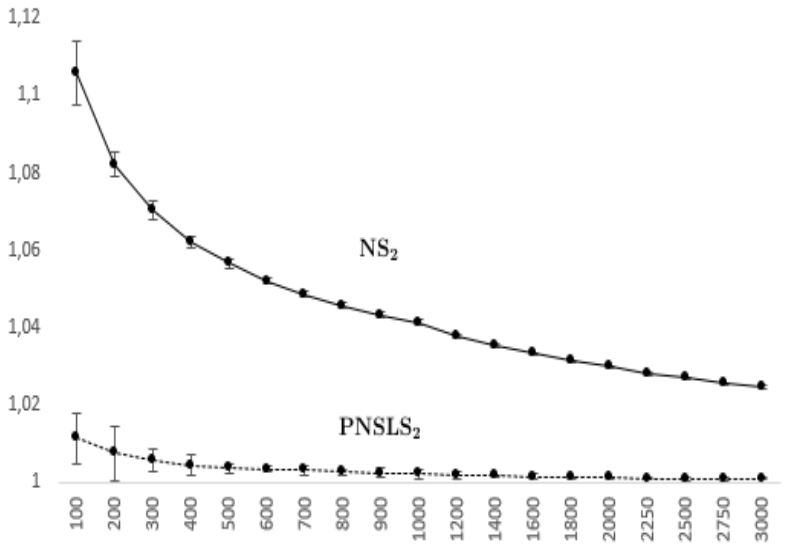

Рис. 3.11: Средние значения случайных величин $d_{\mathbf{NS_2}}(n)$ и $d_{\mathbf{PNSLS_2}}(n)$ при $p = 0.5$.

При $p = 0.67$ относительная средняя ошибка алгоритма $\mathbf{NS_2}$ растет, как и на графах малой размерности. Относительная средняя ошибка алгоритма $\mathbf{PNSLS_2}$ равняется 0, поскольку алгоритмы $\mathbf{PNSLS_2}$ и $\mathbf{NSLS_2}$ дают одинаковые решения (рис. 3.12).

По результатам вычислительного эксперимента можно говорить о том, что достаточных оснований для отклонения гипотез 3.5 и 3.6 нет. Интересно отметить то, что алгоритм $\mathbf{PNSLS_2}$, гарантированная оценка точности которого не доказана, дает решения, которые не слишком сильно уступают по точности решениям алгоритма $\mathbf{NSLS_2}$. При этом он достаточно быстр (незначительно уступая алгоритму $\mathbf{NS_2}$). Это может свидетельствовать о том, что использование процедуры локального поиска к некоторому допустимому решению значительно улучшает качество этого решения.

Полные сведения о результатах эксперимента на графах большей размерности приведены в таблицах 3.23 – 3.33 приложения.

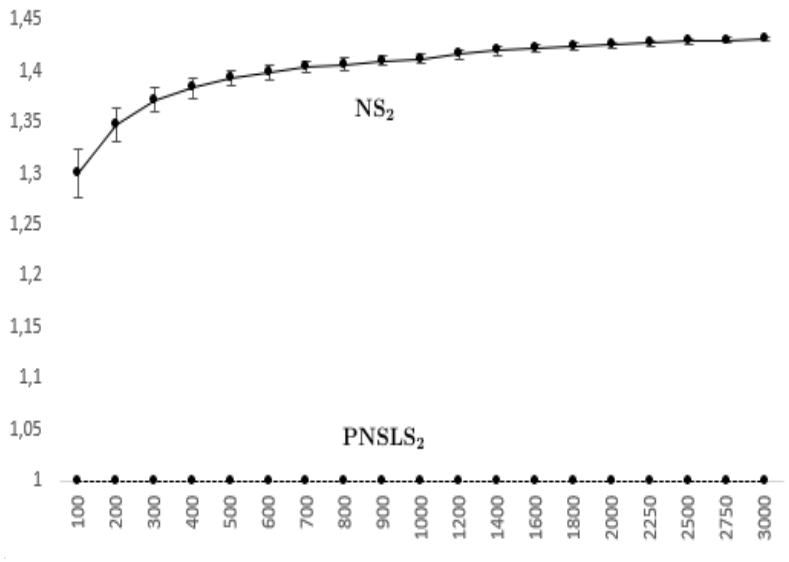

Рис. 3.12: Средние значения случайных величин $d_{\mathbf{NS_2}}(n)$ и $d_{\mathbf{PNSLS_2}}(n)$ при $p = 0.67$.

Экспериментальное исследование алгоритмов для задачи $\mathbf{GC_{\leqslant 3}}$

Наконец, в последнем эксперименте будут исследованы алгоритмы $\mathbf{NLS_{\leqslant 3}}$, $\mathbf{DN_{\leqslant 3}}$, а также алгоритм $\mathbf{DN1LS_{\leqslant 3}}$ (Double neighborhood with one Local Search for $\mathbf{SGC_{\leqslant 3}}$), являющемуся модификацией алгоритма $\mathbf{DN_{\leqslant 3}}$ (к допустимому решению, полученному алгоритмом $\mathbf{DN_{\leqslant 3}}$, была применена процедура локального поиска). Этот алгоритм сохраняет гарантированную оценку точности алгоритма $\mathbf{DN_{\leqslant 3}}$.

Был проведен предварительный эксперимент на графах размерности от 15 до 35 вершин из пространства $G(n, p)$. Для таких графов алгоритмом **BBM**, алгоритмом полного перебора **BF** и с помощью решателей **Gurobi** и **CBC**, использующих модель целочисленного линейного программирования (3.6) – (3.9), удалось найти точные решения. Это позволило сделать вывод о времени работы точных алгоритмов, а также сделать предположения об изменении точности алгоритмов в зависимости от параметров n и p.

Время работы точных алгоритмов (таблицы 3.34 – 3.36 приложения) распределилось следующим образом. Худшие результаты принадлежат решателю **CBC**. Его время работы при $n = 15$ и $p = 0.5$ было равно 717.42 сек, что хуже показателей алгоритма полного перебора **BF**, время работы которого составило 493.85 сек при $n = 20$. Решатель **Gurobi** справился с задачей в случае $n = 25$ за 2324.65 сек. Наилучшие результаты вновь принадлежат ал-

горитму **BBM**, который решил задачу для графов с 35 вершинами в среднем за 2402.47 сек.

На графах малой размерности средняя ошибка алгоритмов $\mathbf{DN}_{\leqslant \mathbf{3}}$ и $\mathbf{NLS}_{\leqslant \mathbf{3}}$ не убывала вне зависимости от значения параметра p. Средняя ошибка алгоритма $\mathbf{DN1LS}_{\leqslant \mathbf{3}}$ была около 0 (рис. 3.13). Поэтому, наиболее перспективным можно считать алгоритм $\mathbf{DN1LS}_{\leqslant \mathbf{3}}$.

Для дальнейшего исследования были введены две случайные величины:

$$d_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n) = \frac{\delta_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n)}{\delta_{\mathbf{DN1LS}_{\leqslant \mathbf{3}}}(n)},\ d_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n) = \frac{\delta_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n)}{\delta_{\mathbf{DN1LS}_{\leqslant \mathbf{3}}}(n)}.$$

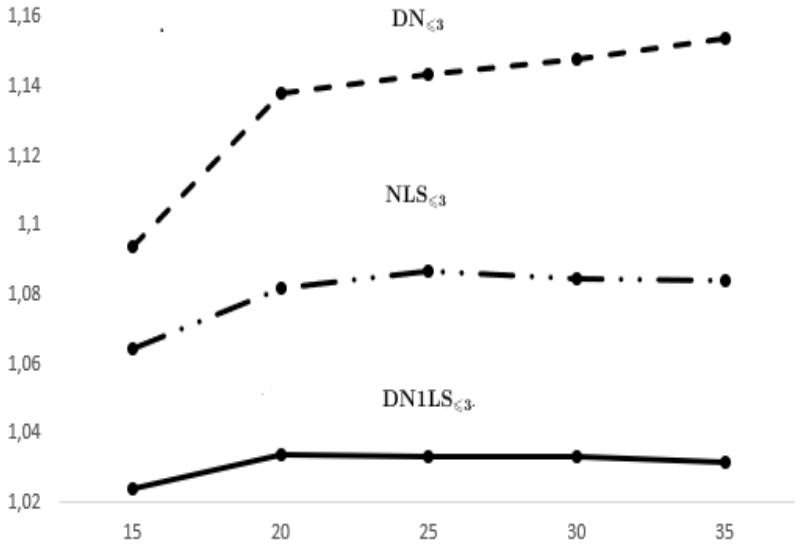

Рис. 3.13: Средняя точность алгоритмов $\mathbf{DN}_{\leqslant \mathbf{3}}$, $\mathbf{NLS}_{\leqslant \mathbf{3}}$ и $\mathbf{DN1LS}_{\leqslant \mathbf{3}}$ на графах малой размерности при $p = 0.33$.

Были выдвинуты следующие гипотезы, проверка которых осуществлена на графах большей размерности.

Гипотеза 3.7. *При достаточно больших n математическое ожидание случайной величины $d_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n)$ не меньше 1.*

Гипотеза 3.8. *При достаточно больших n математическое ожидание случайной величины $d_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n)$ не меньше 1.*

Для проверки гипотез был проведен эксперимент на графах $G(n, p)$ большей размерности (при n от 100 до 500). На основе выборочных данных были

получены оценки $\overline{d}_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n)$ и $\overline{d}_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n)$ математического ожидания для каждой из исследуемых случайных величин, после чего при уровне значимости $\alpha = 0.05$ расчитаны границы доверительных интервалов.

Для расчета границ доверительного интервала использовался квантиль нормального распределения. Статистическое обоснование этому было получено путем проверки критерия согласия Колмогорова-Смирнова эмпирической функции распределения $F_{\mathbf{DN}_{\leqslant \mathbf{3}}}(x)$ с соответствующей ей функцией нормального распределения, имеющей параметры $\overline{d}_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n)$ и $\overline{\sigma}_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n)$. То же самое было сделано для случайной величины $d_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n)$.

В таблицах 3.40 и 3.44 приложения приведены значения статистик Колмогорова–Смирнова при различных n и p. Максимум из этих значений достигается при $n = 350$ и $p = 0.5$ и равно 1.15. Это меньше, чем 1.36 – квантиль распределения Колмогорова-Смирнова при уровне значимости 0.05, что позволяет принять гипотезу о нормальности распределений случайных величин $d_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n)$ и $d_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n)$.

На графах большей размерности поведение алгоритмов $\mathbf{DN}_{\leqslant \mathbf{3}}$ и $\mathbf{NLS}_{\leqslant \mathbf{3}}$ начало отличаться от поведения на графах малой размерности и разделилось на два случая $p \in \{0.33, 0.5\}$ и $p = 0.67$. Поведение алгоритма $\mathbf{DN1LS}_{\leqslant \mathbf{3}}$ сохранило ту же тенденцию.

При $p \in \{0.33, 0.5\}$ относительная ошибка алгоритмов $\mathbf{DN}_{\leqslant \mathbf{3}}$ и $\mathbf{NLS}_{\leqslant \mathbf{3}}$ падает (рис 3.14).

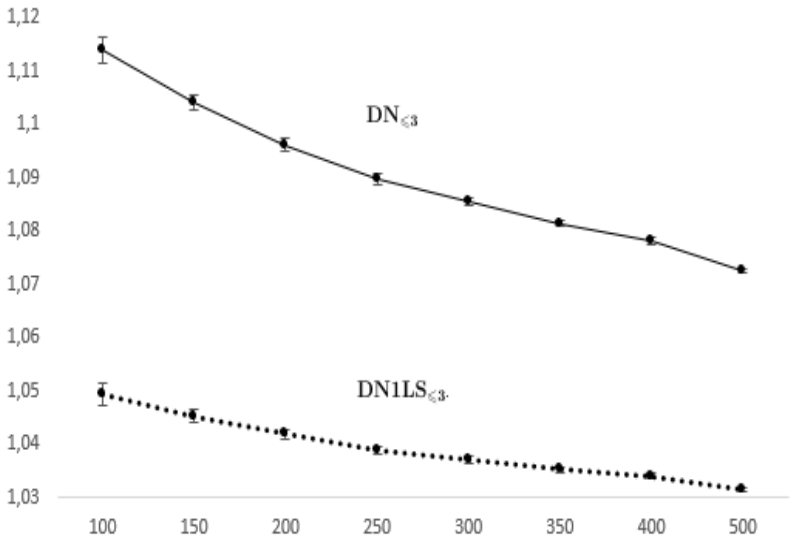

Рис. 3.14: Средние значения случайных величин $d_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n)$ и $d_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n)$ при $p = 0.33$.

При $p = 0.67$ относительная ошибка обоих алгоритмом растет (рис. 3.15).

По результатам вычислительного эксперимента можно говорить о том, что достаточных оснований для отклонения гипотез 3.7 и 3.8 нет. Таким образом, наиболее перспективным алгоритмом является алгоритм $\mathbf{DN1LS_{\leqslant 3}}$. Время его работы незначительно увеличивается по отношению к алгоритму $\mathbf{DN_{\leqslant 3}}$ (при $n = 500$ и $p = 0.5$ 35.5 сек у алгоритма $\mathbf{DN_{\leqslant 3}}$ и 36.41 сек у алгоритма $\mathbf{DN1LS_{\leqslant 3}}$).

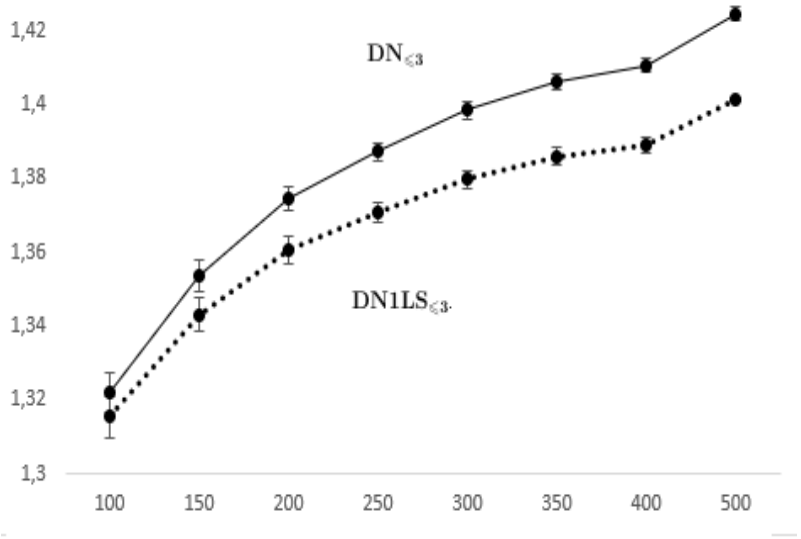

Рис. 3.15: Средние значения случайных величин $d_{\mathbf{DN_{\leqslant 3}}}(n)$ и $d_{\mathbf{NLS_{\leqslant 3}}}(n)$ при $p = 0.67$.

Полные сведения о результатах эксперимента на графах большой размерности приведены в таблицах 3.34 – 3.44 приложения.

3.2.3 Общий вывод по результатам экспериментального исследования

По результатам всех экспериментов можно сделать единые выводы:

1. Алгоритмы, не использующие локальный поиск, имеют плохую точность на плотных графах.

2. Алгоритмы, многократно использующие процедуру локального поиска, находят решения, очень близкие или совпадающие с оптимальными. При этом время работы таких алгоритмов достаточно быстро растет.

3. Алгоритмы, однократно использующие локальный поиск, находят решения, достаточно близкие к оптимальным. При этом время их работы растет не очень быстро.

4. Все алгоритмы имеют наибольшее время работы при значении $p = 0.5$.

Из этого следует, что использование алгоритмов с однократным вызовом процедуры локального поиска является наиболее перспективным с практической точки зрения.

Заключение

В диссертационной работе исследуются различные варианты задач кластеризации вершин графа: задача кластеризации вершин графа с ограниченным числом кластеров, задача кластеризации вершин графа с фиксированным числом кластеров, а также задачи и методы кластеризации вершин графа с частичным обучением.

Основные результаты работы.

1. Для задачи кластеризации вершин графа, в которой число кластеров не превосходит 3, предложены два полиномиальных приближенных алгоритма. Получены априорные гарантированные оценки точности этих алгоритмов.

2. Для задачи кластеризации вершин графа на два кластера разработана процедура локального поиска и два полиномиальных приближенных алгоритма с гарантированными оценками точности.

3. Предложены приближенные алгоритмы с частичным обучением для нового варианта задачи кластеризации вершин графа, в котором число кластеров равно 2. Получены априорные гарантированные оценки точности этих алгоритмов.

4. Предложены два точных метода решения различных вариантов задачи кластеризации вершин графа. Первый использует идею метода ветвей и границ, а второй опирается на известные и новые модели целочисленного линейного программирования рассматриваемых задач. На сериях случайных графов проведено сравнение среднего времени работы точных методов, а также экспериментальное исследование качества решений, найденных рассмотренными в работе приближенными алгоритмами.

Литература

[1] Агеев А.А., Ильев В.П., Кононов А.В., Талевнин А.С. Вычислительная сложность задачи аппроксимации графов // Дискретный анализ и исследование операций. Сер. 1. 2006. Т. 13. N 1. С. 3–15.

[2] Боровков А.А. К вероятностной постановке двух экономических задач // Доклады АН СССР. 1962. Т. 1. N 5. С. 983–986.

[3] Вейнер Г.А. Об аппроксимакции симметричного рефлексивного бинарного отношения отношением эквивалентности // Труды Таллинского политех. института. Сер. А. 1971. N 313. С. 45–49.

[4] Гимади Э.Х., Глебов Н.И., Перепелица В.А. Алгоритмы с оценками для задач дискретной оптимизации // Проблемы кибернетики. М.: Наука. 1975. Вып. 31. С. 35–42.

[5] Зыков А.А. Основы теории графов // М.: Наука. 1987.

[6] Ильев В.П., Ильева С.Д., Навроцкая А.А. Приближенные алгоритмы для задач аппроксимации графов // Дискретный анализ и исследование операций. 2011. Т. 18. N 1. С. 41–60.

[7] Ильев В.П., Навроцкая А.А., Талевнин А.С. Полиномиальная приближенная схема для задачи аппроксимации неплотных графов // Вестник Омского университета. 2007. Вып. 4. С. 24–27.

[8] Ильев В.П., Фридман Г.Ш. К задаче аппроксимации графами с фиксированным числом компонент // Доклады АН СССР. 1982. Т. 264. N 3. С. 533–538.

[9] Ляпунов А.А. О строении и эволюции управляющих систем в связи с теорией классификации // Проблемы кибернетики. М.: Наука. 1973. Вып. 27. С. 7–18.

[10] Миркин Б.Г. Задачи аппроксимации в пространстве отношений и анализ нечисловых данных // Автоматика и телемеханика. 1974. N 9. С. 59–61.

[11] Навроцкая А.А., Ильев В.П., Талевнин А.С. Асимптотически точный алгоритм для задачи аппроксимации неплотных графов // Материалы III Всероссийской конференции «Проблемы оптимизации и экономические приложения». Омск. 2006. С. 115.

[12] Талевнин А.С. О сложности задачи аппроксимации графов// Вестник Омского университета. 2004. N 4. С. 22—24.

[13] Тышкевич Р.И. Матроидные разложения графа // Дискретная математика. 1989. Т. 1. вып. 3. С. 129–139.

[14] Фридман Г.Ш. Одна задача аппроксимации графов // Управляемые системы. Сборник научных трудов. Новосибирск: Институт математики СО АН СССР. 1971. Вып. 8. С. 73–75.

[15] Фридман Г.Ш. Об одном неравенстве в задаче аппроксимации графов // Кибернетика. 1974. N 3. С.151.

[16] Фридман Г.Ш. О некоторых результатах в задаче аппроксимации графов // Проблемы анализа дискретной информации. Новосибирск: ИЭиООП СО АН СССР. 1975. С. 125–152.

[17] Фридман Г.Ш. Исследование одной задачи классификации на графах // В сборнике: Методы моделирования и обработки информации. Новосибирск: Наука. 1976. С. 147–177.

[18] Фридман Г. Ш. Полиномиальная полнота некоторых модификаций задачи аппроксимации графов // Тезисы докладов IV Всесоюзной конференции по проблемам теоретической кибернетики. Новосибирск. 1977. С. 150.

[19] Эрдеш П., Спенсер Дж. Вероятностные методы в комбинаторике // М.: Мир. 1976.

[20] Ailon N., Charikar M., Newman A. Aggregating inconsistent information: Ranking and clustering // Journal of ACM. 2008. V. 55. N 5. P. 1–27.

[21] Alon N., Spencer J.H. The probabilistic method // New York: Wiley and Sons. 1992.

[22] Bair E. Semi-supervised clustering methods // Wiley Interdisciplinary Reviews: Computational Statistics. 2013. V. 5. N 5. P. 349–361.

[23] Bansal N., Blum A., Chawla Sh. Correlation clustering // Machine Learning. 2004. V. 56. P. 89–113.

[24] Bastos L., Ochi L. S., Protti F., Subramanian A., Martins I. C., Pinheiro R. G. S. Efficient algorithms for cluster editing // Journal of Combinatorial Optimization. 2016. V. 31. P. 347–371.

[25] Ben-Dor A., Shamir R., Yakhimi Z. Clustering gene expression patterns // Journal of Computational Biology. 1999. V. 6. N 3–4. P. 281-–297.

[26] Balamurugan R., Natarajan A.M., Premalatha K. Stellar-Mass Black Hole Optimization for Biclustering Microarray Gene Expression Data // Applied Artificial Intelligence. 2015. V. 29. N 4. P. 353–381.

[27] Bollobas B. The chromatic number of random graphs // Combinatorica. 1988. V. 8. P. 49–55.

[28] Chapelle O., Scholkopf B., Zein A. Semi-supervised learning // MIT Press: Cambridge, Massachusets. 2006.

[29] Charikar M., Guruswami V., Wirth A. Clustering with qualitative information// Journal of Computer and System Sciences. 2005. V. 71. N 3. P. 360–383.

[30] Chawla S., Makarychev K., Schramm T., Yaroslavtsev G. Near optimal LP algorithm for correlation clustering on complete and complete k-partite

graphs // STOC '15 Symposium on Theory of Computing: ACM New York. 2015. P. 219-228.

[31] Chen Z.-Z., Jiang T., Lin G. Computing philogenetic roots with bounded degrees and errors // SIAM Journal on Computing. 2003. V. 32. N 4. P. 864–879.

[32] Coleman T., Saunderson J., Wirth A. A local-search 2-approximation for 2-correlation-clustering // Algorithms - ESA 2008: Lecture Notes in Computer Science. 2008 V. 5193. P. 308–319.

[33] DasGupta B., Enciso G., Sontag E., Zhang Y. Algorithmic and complexity results for decompositions of biological networks into monotone subsystems // BioSystems. 2007. V. 90. N 1. 161-–178.

[34] Giotis I., Guruswami V. Correlation clustering with a fixed number of clusters // Theory of Computing. 2006. V. 2. N 1. P. 249–266.

[35] Harary F. On the notion of balance of a signed graph // Michigan Mathematical Journal. 1953. N 2. P. 143-146.

[36] Kriegel H. P., Kroger P., Zimek A. Clustering high-dimensional data // ACM Transactions on Knowledge Discovery from Data. 2009. N 3. P. 1–58.

[37] Křivánek M., Morávek J. NP-hard problems in hierarchical-tree clustering // Acta informatica. 1986. V. 23. P. 311-323.

[38] Komusiewicz C., Uhlmann J. Cluster editing with locally bounded modi cations // Discrete Applied Mathematics. 2012. V. 160. N 15. P. 2259–2270.

[39] Mannaa B. Cluster editing problem for points on the real line: A polynomial time algorithm // Information Processing Letters. 2010. V. 110. P. 961–965.

[40] Shamir R., Sharan R., Tsur D. Cluster graph modification problems // Discrete Applied Mathematics. 2004. V. 144. N 1–2. P. 173–182.

[41] Sole P., Zaslavsky T. A coding approach to signed graphs // SIAM Journal on Descrete Mathematics. 1994. N 7. P. 544–553.

[42] Tomescu I. Note sur une caractérisation des graphes done le degreé de deséquilibre est maximal // Mathematiques et sciences humaines. 1973. V. 11. N 42. P. 37–40.

[43] Tomescu I. La reduction minimale dún graphe á une reunion de cliques // Discrete Mathematics. 1974. V. 10. N 1–2. P. 173–179.

[44] Xin X. An FPT algorithm for the correlation clustering problem // Key Engineering Materials. Advanced Materials and Computer Science. 2011. V. 474–476. P. 924–927.

[45] Zahn C. Approximating symmetric relations by equivalence relations // Journal of SIAM. 1964. V. 12. N 4. P. 840–847.

Публикации автора по теме диссертации

1. Ильев В.П., Ильева С.Д., Моршинин А.В. 2-приближённые алгоритмы для двух задач кластеризации на графах // Дискретный анализ и исследование операций. 2020. Т. 27. N 3. С. 88–108.
2. Ильев В.П., Ильева С.Д., Моршинин А.В. Алгоритмы приближённого решения одной задачи кластеризации графа // Прикладная дискретная математика. 2019. N 45. С. 64–77.
3. Моршинин А.В. Об одной задаче кластеризации графа // Вестник Омского университета. 2018. Т 23. N 1. С. 4–9.
4. Моршинин А.В. Точные алгоритмы для задач кластеризации вершин графа // Вестник Омского университета. 2021. Т. 26. N 2. С. 23–29.
5. Il'ev V., Il'eva S., Morshinin A. A 2-approximation algorithm for the graph 2-clustering problem // In: M. Khachay et al. (Eds.) MOTOR 2019. Lecture Notes in Computer Science. Springer. 2019. V. 11548. P. 295—308.
6. Il'ev V., Il'eva S., Morshinin A. An approximation algorithm for a semi-supervised graph clustering problem // In: Yu. Kochetov et.al. (Eds.) MOTOR 2020. Communications in Computer and Information Science. Springer. 2020. V. 1275. P. 23–29.
7. Моршинин А.В. Приближенное решение одной задачи кластеризации графа // Всероссийская научно-практическая конференция «Омские научные чтения». Материалы конференции. Омск 2017. С. 1041–1043.
8. Моршинин А.В. Метод ветвей и границ для задач кластеризации вершин графа // Четвертая всероссийская научно-практическая конференция «Омские научные чтения». Материалы конференции. Омск 2020. С. 2161–2165.

9. Моршинин А.В. Алгоритм приближенного решения одной задачи кластеризации графа // IV региональная конференция магистрантов, аспирантов и молодых ученых по физике, математике и химии «ФМХ ОмГУ 2016». Сборник статей конференции. Омск 2016. С. 15–18.

10. Моршинин А.В. Приближенное решение задачи кластеризации графа // V региональная конференция магистрантов, аспирантов и молодых ученых по физике, математике и химии «ФМХ ОмГУ 2017». Сборник статей конференции. Омск 2017. С. 15–18.

11. Ильев В.П., Ильева С.Д., Моршинин А.В. Одна задача кластеризации с частичным обучением // VII Международная конференция «Проблемы оптимизации и их приложения (OPTA 2018)». Тезисы докладов конференции. Омск 2018. С. 85.

Приложения

Данные экспериментального исследования алгоритмов для задачи $\mathbf{GC_{\leqslant 2}}$

Таблица 3.1: Среднее время работы алгоритмов в секундах при $p = 0.33$

n	**BF**	**BBM**	**CBC**	**Gurobi**	**BBC**	**CSW**	**N1LS$_{\leqslant 2}$**
15	0	0	5.5	0	0	0	0
20	2	0	70.3	2.1	0	0	0
25	18.5	0	1300.54	15.5	0	0	0
30	875.8	0	-	135.96	0	0	0
35	-	0	-	1700.6	0	0	0
40	-	2.21	-	-	0	0	0
50	-	262.72	-	-	0	0	0
100	-	-	-	-	0	0	0
200	-	-	-	-	0	0	0
300	-	-	-	-	0	0	0
400	-	-	-	-	0	0	0
500	-	-	-	-	0	0	0
600	-	-	-	-	0	0	0
700	-	-	-	-	0	0	0
800	-	-	-	-	0	0.07	0
900	-	-	-	-	0	1	0
1000	-	-	-	-	0	1.11	0
1200	-	-	-	-	0.16	3	0.38
1400	-	-	-	-	1	4.71	1
1600	-	-	-	-	2	7	2
1800	-	-	-	-	3	10.13	3
2000	-	-	-	-	4	15.25	4
2250	-	-	-	-	5.96	22.5	6.01
2500	-	-	-	-	8.01	31.31	8
2750	-	-	-	-	10.58	42.14	10.99
3000	-	-	-	-	14.37	58.95	14.44
3500	-	-	-	-	23.42	96.83	23.73
4000	-	-	-	-	35.28	145.57	35.39
5000	-	-	-	-	66.71	280.66	67.25
6000	-	-	-	-	115.25	493.2	115.89

Таблица 3.2: Среднее время работы алгоритмов в секундах при $p = 0.5$

n	**BF**	**BBM**	**CBC**	**Gurobi**	**BBC**	**CSW**	**N1LS$_{\leqslant 2}$**
15	0	0	10.3	0	0	0	0
20	2	0	72.2	3.31	0	0	0
25	18.5	0	1404.4	15.8	0	0	0
30	875.8	0	-	150.2	0	0	0
35	-	0.95	-	2000.6	0	0	0
40	-	15.31	-	-	0	0	0
50	-	1781.96	-	-	0	0	0
100	-	-	-	-	0	0	0
200	-	-	-	-	0	0	0
300	-	-	-	-	0	0	0
400	-	-	-	-	0	0	0
500	-	-	-	-	0	0	0
600	-	-	-	-	0	0	0
700	-	-	-	-	0	0	0
800	-	-	-	-	0	0.78	0
900	-	-	-	-	0	1	0
1000	-	-	-	-	0.01	1.11	0
1200	-	-	-	-	1	3.01	1
1400	-	-	-	-	1	5	1
1600	-	-	-	-	2	8	2
1800	-	-	-	-	3.05	11.04	3
2000	-	-	-	-	5.02	17.31	5.07
2250	-	-	-	-	7.14	24.45	7.12
2500	-	-	-	-	10.02	33.41	10
2750	-	-	-	-	13.03	44.98	13.03
3000	-	-	-	-	17	60.13	17.01
3500	-	-	-	-	27.42	99.31	27.49
4000	-	-	-	-	40.7	150.08	41.06
5000	-	-	-	-	81.76	305.32	82.05
6000	-	-	-	-	149.21	580.68	149.58

Таблица 3.3: Среднее время работы алгоритмов в секундах при $p = 0.67$

n	**BF**	**BBM**	**CBC**	**Gurobi**	**BBC**	**CSW**	**N1LS$_{\leqslant 2}$**
15	0	0	1	0	0	0	0
20	2	0	50	0.33	0	0	0
25	18.5	0	1010.89	5.5	0	0	0
30	875.8	0	-	56.7	0	0	0
35	-	0.08	-	1057.3	0	0	0
40	-	2.78	-	-	0	0	0
50	-	110.03	-	-	0	0	0
100	-	-	-	-	0	0	0
200	-	-	-	-	0	0	0
300	-	-	-	-	0	0	0
400	-	-	-	-	0	0	0
500	-	-	-	-	0	0	0
600	-	-	-	-	0	0	0
700	-	-	-	-	0	0	0
800	-	-	-	-	0	0.78	0
900	-	-	-	-	0	0.34	0
1000	-	-	-	-	0	1	0
1200	-	-	-	-	0.47	2	0.6
1400	-	-	-	-	1	3	1
1600	-	-	-	-	2	5	2
1800	-	-	-	-	3	7	3
2000	-	-	-	-	4.03	10.36	4.05
2250	-	-	-	-	6.05	15.03	6.02
2500	-	-	-	-	8.05	20.14	8.09
2750	-	-	-	-	11.28	27.97	11.28
3000	-	-	-	-	14.01	37.19	14
3500	-	-	-	-	23.23	62.48	23.3
4000	-	-	-	-	33.43	90.34	33.67
5000	-	-	-	-	68.43	184.98	68.76
6000	-	-	-	-	115.57	314.71	115.98

Таблица 3.4: Среднее значение $\overline{d}_{\mathbf{BBC}}(n)$ по выборке случайной величины $d_{\mathbf{BBC}}(n) = \delta_{\mathbf{BBC}}(n)/\delta_{\mathbf{CSW}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	1.122438	1.109220	1.319653
200	1.099946	1.083810	1.361766
300	1.088085	1.071263	1.380263
400	1.080558	1.062887	1.389788
500	1.075646	1.056574	1.398440
600	1.072151	1.052041	1.402047
700	1.069305	1.048623	1.405849
800	1.067139	1.045777	1.409519
900	1.064910	1.043280	1.412306
1000	1.063577	1.041365	1.414317
1200	1.060959	1.038044	1.418443
1400	1.058868	1.035337	1.421239
1600	1.057339	1.033158	1.423145
1800	1.056063	1.031396	1.424849
2000	1.055321	1.029914	1.426457
2250	1.054028	1.028246	1.427791
2500	1.053062	1.026889	1.429452
2750	1.052392	1.025687	1.430699
3000	1.051695	1.024678	1.431729
3500	1.050441	1.022924	1.433208
4000	1.049573	1.021498	1.434730
5000	1.048297	1.019318	1.436587
6000	1.047380	1.017691	1.438422

Таблица 3.5: Стандартное отклонение $\overline{\sigma}_{\mathbf{BBC}}(n)$ по выборке случайной величины $d_{\mathbf{BBC}}(n) = \delta_{\mathbf{BBC}}(n)/\delta_{\mathbf{CSW}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.010634	0.007588	0.025102
200	0.004695	0.003990	0.014857
300	0.003679	0.002581	0.012555
400	0.002977	0.001650	0.009190
500	0.002680	0.001296	0.008105
600	0.002199	0.001111	0.007239
700	0.002599	0.000986	0.006830
800	0.001899	0.000761	0.005992
900	0.002094	0.000721	0.005877
1000	0.002027	0.000508	0.004727
1200	0.001732	0.000431	0.003771
1400	0.001264	0.000429	0.004190
1600	0.001520	0.000320	0.003903
1800	0.001406	0.000293	0.003026
2000	0.001302	0.000294	0.003077
2250	0.001279	0.000221	0.003517
2500	0.001301	0.000229	0.002424
2750	0.001078	0.000201	0.002248
3000	0.001036	0.000206	0.002310
3500	0.001082	0.000142	0.002352
4000	0.001040	0.000117	0.001891
5000	0.000878	0.000097	0.001984
6000	0.000861	0.000090	0.001553

Таблица 3.6: Границы доверительного интервала для случайной величины $d_{\mathbf{BBC}}(n) = \delta_{\mathbf{BBC}}(n)/\delta_{\mathbf{CSW}}(n)$ при уровне значимости 0.05.

$n \setminus p$	0.33	0.5	0.67
100	[1.120354, 1.124523]	[1.107732, 1.110707]	[1.314733, 1.324574]
200	[1.099026, 1.100867]	[1.083028, 1.084592]	[1.358854, 1.364678]
300	[1.087364, 1.088806]	[1.070757, 1.071769]	[1.377802, 1.382724]
400	[1.079974, 1.081141]	[1.062563, 1.063210]	[1.387986, 1.391589]
500	[1.075121, 1.076172]	[1.056320, 1.056828]	[1.396851, 1.400028]
600	[1.071720, 1.072583]	[1.051824, 1.052259]	[1.400629, 1.403466]
700	[1.068796, 1.069815]	[1.048429, 1.048816]	[1.404510, 1.407188]
800	[1.066766, 1.067511]	[1.045628, 1.045927]	[1.408344, 1.410693]
900	[1.064499, 1.065320]	[1.043138, 1.043421]	[1.411154, 1.413458]
1000	[1.063180, 1.063974]	[1.041265, 1.041465]	[1.413391, 1.415244]
1200	[1.060619, 1.061299]	[1.037959, 1.038128]	[1.417703, 1.419182]
1400	[1.058620, 1.059116]	[1.035253, 1.035421]	[1.420417, 1.422060]
1600	[1.057041, 1.057637]	[1.033095, 1.033221]	[1.422380, 1.423910]
1800	[1.055788, 1.056339]	[1.031338, 1.031454]	[1.424256, 1.425442]
2000	[1.055066, 1.055577]	[1.029857, 1.029972]	[1.425854, 1.427060]
2250	[1.053777, 1.054279]	[1.028203, 1.028289]	[1.427102, 1.428480]
2500	[1.052807, 1.053317]	[1.026844, 1.026934]	[1.428977, 1.429927]
2750	[1.052180, 1.052603]	[1.025648, 1.025727]	[1.430258, 1.431140]
3000	[1.051492, 1.051898]	[1.024638, 1.024719]	[1.431276, 1.432182]
3500	[1.050229, 1.050654]	[1.022896, 1.022952]	[1.432746, 1.433669]
4000	[1.049369, 1.049777]	[1.021475, 1.021521]	[1.434360, 1.435101]
5000	[1.048125, 1.048469]	[1.019299, 1.019337]	[1.436198, 1.436976]
6000	[1.047211, 1.047549]	[1.017674, 1.017709]	[1.438117, 1.438726]

Таблица 3.7: Статистика Колмогорова-Смирнова для эмпирической функции распределения случайной величины $d_{\mathbf{BBC}}(n) = \delta_{\mathbf{BBC}}(n)/\delta_{\mathbf{CSW}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.57	0.79	0.55
200	0.56	0.55	0.52
300	0.7	0.63	0.96
400	1.01	0.73	0.78
500	0.63	0.81	0.56
600	0.6	0.77	1.01
700	0.76	0.77	0.81
800	0.54	0.61	0.53
900	1.29	0.4	0.9
1000	0.56	0.53	0.59
1200	0.69	0.46	0.83
1400	0.68	1.01	0.87
1600	1	0.63	0.79
1800	0.8	0.47	0.49
2000	0.57	0.91	0.81
2250	0.97	0.55	0.68
2500	0.53	0.71	0.95
2750	0.96	0.71	0.79
3000	0.74	0.52	1.22
3500	0.7	0.63	0.88
4000	0.92	0.55	0.72
5000	1.32	0.47	0.91
6000	0.98	0.75	0.96

Таблица 3.8: Среднее значение $\overline{d}_{\mathbf{N1LS}_{\leqslant \mathbf{2}}}(n)$ по выборке случайной величины $d_{\mathbf{N1LS}_{\leqslant \mathbf{2}}}(n) = \delta_{\mathbf{N1LS}_{\leqslant \mathbf{2}}}(n)/\delta_{\mathbf{CSW}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	1.013105	1.014121	1
200	1.009226	1.009750	1
300	1.006511	1.007302	1
400	1.005475	1.005429	1
500	1.004664	1.004417	1
600	1.003680	1.003852	1
700	1.003413	1.003502	1
800	1.003267	1.003196	1
900	1.002567	1.002901	1
1000	1.002547	1.002564	1
1200	1.002050	1.002180	1
1400	1.001806	1.001954	1
1600	1.001664	1.001726	1
1800	1.001424	1.001670	1
2000	1.001403	1.001512	1
2250	1.001220	1.001318	1
2500	1.001105	1.001179	1
2750	1.001023	1.001073	1
3000	1.000920	1.001016	1
3500	1.000845	1.000901	1
4000	1.000740	1.000810	1
5000	1.000576	1.000639	1
6000	1.000481	1.000536	1

Таблица 3.9: Стандартное отклонение $\overline{\sigma}_{\mathbf{N1LS_{\leqslant 2}}}(n)$ по выборке случайной величины $d_{\mathbf{N1LS_{\leqslant 2}}}(n) = \delta_{\mathbf{N1LS_{\leqslant 2}}}(n)/\delta_{\mathbf{CSW}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.007577	0.007705	0
200	0.004317	0.004510	0
300	0.002921	0.003086	0
400	0.002180	0.002358	0
500	0.001927	0.001633	0
600	0.001389	0.001375	0
700	0.001330	0.0013218	0
800	0.001015	0.0010018	0
900	0.001017	0.0009645	0
1000	0.000944	0.000986	0
1200	0.000673	0.000767	0
1400	0.000533	0.000667	0
1600	0.000492	0.000549	0
1800	0.000436	0.000484	0
2000	0.000397	0.000409	0
2250	0.000377	0.000397	0
2500	0.000360	0.000342	0
2750	0.000293	0.000377	0
3000	0.000319	0.000308	0
3500	0.000266	0.000273	0
4000	0.000206	0.000257	0
5000	0.000174	0.000195	0
6000	0.000151	0.000170	0

Таблица 3.10: Границы доверительного интервала для случайной величины $d_{\mathbf{N1LS}_{\leqslant \mathbf{2}}}(n) = \delta_{\mathbf{N1LS}_{\leqslant \mathbf{2}}}(n)/\delta_{\mathbf{CSW}}(n)$ при уровне значимости 0.05.

$n \setminus p$	0.33	0.5	0.67
100	[1.011620 , 1.014590]	[1.012611 , 1.015631]	[1 , 1]
200	[1.008379 , 1.010072]	[1.008866 , 1.010634]	[1 , 1]
300	[1.005938 , 1.007083]	[1.006697 , 1.007906]	[1 , 1]
400	[1.005048 , 1.005903]	[1.004967 , 1.005891]	[1 , 1]
500	[1.004286 , 1.005041]	[1.004097 , 1.004737]	[1 , 1]
600	[1.003407 , 1.003952]	[1.003582 , 1.004121]	[1 , 1]
700	[1.003153 , 1.003674]	[1.003243 , 1.003762]	[1 , 1]
800	[1.003068 , 1.003466]	[1.003000 , 1.003393]	[1 , 1]
900	[1.002367 , 1.002766]	[1.002712 , 1.003090]	[1 , 1]
1000	[1.002362 , 1.002732]	[1.002370 , 1.002757]	[1 , 1]
1200	[1.001918 , 1.002182]	[1.002029 , 1.002330]	[1 , 1]
1400	[1.001701 , 1.001911]	[1.001823 , 1.002085]	[1 , 1]
1600	[1.001567 , 1.001760]	[1.001618 , 1.001834]	[1 , 1]
1800	[1.001339 , 1.001510]	[1.001575 , 1.001765]	[1 , 1]
2000	[1.001325 , 1.001481]	[1.001431 , 1.001592]	[1 , 1]
2250	[1.001146 , 1.001294]	[1.001240 , 1.001396]	[1 , 1]
2500	[1.001034 , 1.001175]	[1.001112 , 1.001246]	[1 , 1]
2750	[1.000965 , 1.001080]	[1.000999 , 1.001147]	[1 , 1]
3000	[1.000857 , 1.000982]	[1.000955 , 1.001076]	[1 , 1]
3500	[1.000793 , 1.000898]	[1.000847 , 1.000954]	[1 , 1]
4000	[1.000700 , 1.000781]	[1.000759 , 1.000860]	[1 , 1]
5000	[1.000542 , 1.000610]	[1.000601 , 1.000677]	[1 , 1]
6000	[1.000451 , 1.000511]	[1.000502 , 1.000569]	[1 , 1]

Таблица 3.11: Статистика Колмогорова-Смирнова для эмпирической функции распределения случайной величины $d_{\mathbf{N1LS_{\leqslant 2}}}(n) = \delta_{\mathbf{N1LS_{\leqslant 2}}}(n)/\delta_{\mathbf{CSW}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.79	0.74	-
200	0.94	0.8	-
300	0.7	0.81	-
400	0.5	0.76	-
500	0.52	0.54	-
600	0.82	0.51	-
700	0.57	0.47	-
800	0.76	0.51	-
900	0.41	0.76	-
1000	0.77	0.69	-
1200	0.57	0.66	-
1400	0.62	0.61	-
1600	0.52	0.73	-
1800	0.87	0.63	-
2000	0.61	0.62	-
2250	0.58	0.38	-
2500	0.48	0.65	-
2750	0.84	0.42	-
3000	0.48	0.5	-
3500	0.55	0.46	-
4000	0.57	0.64	-
5000	0.6	0.81	-
6000	0.74	0.54	-

Данные экспериментального исследования алгоритмов для задачи $\mathbf{GC_2}$

Таблица 3.12: Среднее время работы алгоритмов в секундах при $p = 0.33$

n	**BF**	**BBM**	**CBC**	**Gurobi**	$\mathbf{N_2}$	$\mathbf{NLS_2}$	$\mathbf{N1LS_2}$
10	0	0	0	0	0	0	0
15	0	0	10	0	0	0	0
20	2	0	100.2	5.5	0	0	0
25	18.5	0	1423.78	15.67	0	0	0
30	875.8	0	-	138.4	0	0	0
35	-	0.01	-	1812.12	0	0	0
40	-	1.97	-	-	0	0	0
45	-	24.85	-	-	0	0	0
50	-	259.75	-	-	0	0	0
100	-	-	-	-	0	1.99	0
200	-	-	-	-	0	16.07	0.03
300	-	-	-	-	4.06	49.55	4.1
400	-	-	-	-	13.35	112.52	13.38
500	-	-	-	-	33.05	220.45	33.12
600	-	-	-	-	66.85	370.96	66.99

Таблица 3.13: Среднее время работы алгоритмов в секундах при $p = 0.5$

n	**BF**	**BBM**	**CBC**	**Gurobi**	$\mathbf{N_2}$	$\mathbf{NLS_2}$	$\mathbf{N1LS_2}$
10	0	0	0	0	0	0	0
15	0	0	12	0	0	0	0
20	2	0	131.64	1	0	0	0
25	18.5	0	2182.22	10.1	0	0	0
30	875.8	0	-	150.5	0	0	0
35	-	1.09	-	2150.6	0	0	0
40	-	14.72	-	-	0	0	0
45	-	166.27	-	-	0	0	0
50	-	2003.14	-	-	0	0	0
100	-	-	-	-	0	1.96	0
200	-	-	-	-	0.04	15.52	0.69
300	-	-	-	-	5.01	47.82	5.01
400	-	-	-	-	15.68	106.45	15.73
500	-	-	-	-	39.74	207.85	39.83
600	-	-	-	-	80.27	343.67	80.54

Таблица 3.14: Среднее время работы алгоритмов в секундах при $p = 0.67$

n	**BF**	**BBM**	**CBC**	**Gurobi**	$\mathbf{N_2}$	$\mathbf{NLS_2}$	$\mathbf{N1LS_2}$
10	0	0	0	0	0	0	0
15	0	0	0	15	0	0	0
20	2	0	100.2	3	0	0	0
25	18.5	0	1320	15.4	0	0	0
30	875.8	0	-	30	0	0	0
35	-	0.09	-	612.4	0	0	0
40	-	2.99	-	-	0	0	0
45	-	18.35	-	-	0	0	0
50	-	112.6	-	-	0	0	0
100	-	-	-	-	0	1.04	0
200	-	-	-	-	0.06	13.56	0.07
300	-	-	-	-	4	44	4
400	-	-	-	-	13.27	87.22	13.3
500	-	-	-	-	34.25	165.62	34.34
600	-	-	-	-	69.26	279.39	69.26

Таблица 3.15: Среднее значение $\overline{d}_{\mathbf{N_2}}(n)$ по выборке случайной величины $d_{\mathbf{N_2}}(n) = \delta_{\mathbf{N_2}}(n)/\delta_{\mathbf{NLS_2}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	1.116219	1.099872	1.301999
200	1.100336	1.081449	1.351769
300	1.088967	1.070357	1.373808
400	1.081675	1.062617	1.385154
500	1.076630	1.057354	1.394059
600	1.073363	1.052792	1.398970

Таблица 3.16: Стандартное отклонение $\overline{\sigma}_{\mathbf{N_2}}(n)$ по выборке случайной величины $d_{\mathbf{N_2}}(n) = \delta_{\mathbf{N_2}}(n)/\delta_{\mathbf{N1LS_2}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.009746	0.007713	0.026596
200	0.004973	0.003955	0.016272
300	0.004200	0.003138	0.010994
400	0.003015	0.002322	0.011132
500	0.002692	0.001786	0.007997
600	0.002418	0.001478	0.008804

Таблица 3.17: Границы доверительного интервала для случайной величины $d_{\mathbf{N_2}}(n) = \delta_{\mathbf{N_2}}(n)/\delta_{\mathbf{NLS_2}}(n)$ при уровне значимости 0.05.

$n \setminus p$	0.33	0.5	0.67
100	[1.1143096, 1.1181301]	[1.098431, 1.101314]	[1.296786, 1.307212]
200	[1.0993612, 1.1013109]	[1.080754, 1.082145]	[1.348579, 1.354958]
300	[1.0881440, 1.0897906]	[1.070007, 1.070707]	[1.371653, 1.375963]
400	[1.0810846, 1.0822668]	[1.062295, 1.062940]	[1.382971, 1.387336]
500	[1.0761026, 1.0771579]	[1.057110, 1.057598]	[1.392492, 1.395627]
600	[1.0728899, 1.0738378]	[1.052590, 1.052994]	[1.397245, 1.400696]

Таблица 3.18: Статистика Колмогорова-Смирнова для эмпирической функции распределения случайной величины $d_{\mathbf{N_2}}(n) = \delta_{\mathbf{N_2}}(n)/\delta_{\mathbf{NLS_2}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	1.25	0.85	0.95
200	0.76	0.89	0.62
300	0.64	1.17	0.65
400	0.83	0.43	0.83
500	0.49	0.51	0.54
600	0.82	0.87	0.94

Таблица 3.19: Среднее значение $\overline{d}_{\mathbf{N1LS_2}}(n)$ по выборке случайной величины $d_{\mathbf{N1LS_2}}(n) = \delta_{\mathbf{N1LS_2}}(n)/\delta_{\mathbf{NLS_2}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	1.014346	1.015938	1.008225
200	1.010465	1.011341	1.004160
300	1.008836	1.008674	1.002814
400	1.007287	1.007720	1.001914
500	1.006156	1.005986	1.001612
600	1.004979	1.005641	1.001201

Таблица 3.20: Стандартное отклонение $\overline{\sigma}_{\mathbf{N1LS_2}}(n)$ по выборке случайной величины $d_{\mathbf{N1LS_2}}(n) = \delta_{\mathbf{N1LS_2}}(n)/\delta_{\mathbf{N1LS_2}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.007459	0.007713	0.005754
200	0.004068	0.003955	0.002247
300	0.002928	0.003138	0.001348
400	0.002213	0.002322	0.000726
500	0.001613	0.001786	0.000524
600	0.001387	0.001478	0.000473

Таблица 3.21: Границы доверительного интервала для случайной величины $d_{\mathbf{N1LS_2}}(n) = \delta_{\mathbf{N1LS_2}}(n)/\delta_{\mathbf{NLS_2}}(n)$ при уровне значимости 0.05.

$n \setminus p$	0.33	0.5	0.67
100	[1.012884, 1.015808]	[1.014426, 1.0174504]	[1.007097, 1.009353]
200	[1.009668, 1.011263]	[1.010566, 1.0121167]	[1.003719, 1.004600]
300	[1.008262, 1.009410]	[1.008059, 1.0092901]	[1.002550, 1.003079]
400	[1.006853, 1.007720]	[1.007264, 1.0081752]	[1.001772, 1.002056]
500	[1.005840, 1.006472]	[1.005636, 1.0063369]	[1.001509, 1.001714]
600	[1.004707, 1.005251]	[1.005351, 1.0059310]	[1.001108, 1.001293]

Таблица 3.22: Статистика Колмогорова-Смирнова для эмпирической функции распределения случайной величины $d_{\mathbf{N1LS_2}}(n) = \delta_{\mathbf{N1LS_2}}(n)/\delta_{\mathbf{NLS_2}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.73	0.58	0.95
200	0.57	0.83	0.6
300	0.99	0.89	1.06
400	0.58	0.89	0.7
500	0.45	0.95	0.72
600	0.97	0.77	0.54

Данные экспериментального исследования алгоритмов для задачи $\mathbf{SGC_2}$

Таблица 3.23: Среднее время работы алгоритмов в секундах при $p = 0.33$

n	**BF**	**BBM**	**CBC**	**Gurobi**	$\mathbf{NS_2}$	$\mathbf{NSLS_2}$	$\mathbf{PNSLS_2}$
15	0	0	3	0	0	0	0
20	2	0	20.5	0.5	0	0	0
25	18.5	0	680.5	2	0	0	0
30	875.8	0	-	25.45	0	0	0
35	-	0	-	512.87	0	0	0
40	-	0.99	-	-	0	0	0
45	-	16.87	-	-	0	0	0
50	-	139.04	-	-	0	0	0
100	-	-	-	-	0	0	0
200	-	-	-	-	0	0	0
300	-	-	-	-	0	0	0
400	-	-	-	-	0	0	0
500	-	-	-	-	0	0.07	0
600	-	-	-	-	0	1	0
700	-	-	-	-	0	1.04	0
800	-	-	-	-	0	2	0
900	-	-	-	-	0	3	0
1000	-	-	-	-	1	4	1
1200	-	-	-	-	1	6.65	1
1400	-	-	-	-	2.21	10.73	2.32
1600	-	-	-	-	4.01	16.45	4.02
1800	-	-	-	-	5.99	25.32	6
2000	-	-	-	-	8.04	32.84	8.04
2250	-	-	-	-	11.75	49.53	11.76
2500	-	-	-	-	15.49	66.45	15.78
2750	-	-	-	-	21.45	91.34	21.57
3000	-	-	-	-	27.65	119.39	27.78

Таблица 3.24: Среднее время работы алгоритмов в секундах при $p = 0.5$

n	**BF**	**BBM**	**CBC**	**Gurobi**	$\mathbf{NS_2}$	$\mathbf{NSLS_2}$	$\mathbf{PNSLS_2}$
15	0	0	4	0	0	0	0
20	2	0	30	0.5	0	0	0
25	18.5	0	700.4	4	0	0	0
30	875.8	0	-	30.6	0	0	0
35	-	0.31	-	601.8	0	0	0
40	-	8.61	-	-	0	0	0
45	-	107.39	-	-	0	0	0
50	-	1041.37	-	-	0	0	0
100	-	-	-	-	0	0	0
200	-	-	-	-	0	0	0
300	-	-	-	-	0	0	0
400	-	-	-	-	0	0	0
500	-	-	-	-	0	0	0
600	-	-	-	-	0	1	0
700	-	-	-	-	0	1.17	0
800	-	-	-	-	0	2	0
900	-	-	-	-	0.07	3.01	0.1
1000	-	-	-	-	1	4.02	1
1200	-	-	-	-	2	7	2
1400	-	-	-	-	3	11.22	3.02
1600	-	-	-	-	5.04	18.51	5.02
1800	-	-	-	-	7.07	25.44	7.1
2000	-	-	-	-	10.01	36.25	10.01
2250	-	-	-	-	14.2	51.57	14.21
2500	-	-	-	-	20.18	74.85	20.18
2750	-	-	-	-	26.25	98.13	26.31
3000	-	-	-	-	33.88	126.03	33.92

Таблица 3.25: Среднее время работы алгоритмов в секундах при $p = 0.67$

n	**BF**	**BBM**	**CBC**	**Gurobi**	$\mathbf{NS_2}$	$\mathbf{NSLS_2}$	$\mathbf{PNSLS_2}$
15	0	0	0	0	0	0	0
20	2	0	2	0	0	0	0
25	18.5	0	10.5	0.5	0	0	0
30	875.8	0	-	3	0	0	0
35	-	0.11	-	9	0	0	0
40	-	2.51	-	-	0	0	0
45	-	17.97	-	-	0	0	0
50	-	93.31	-	-	0	0	0
100	-	-	-	-	0	0	0
200	-	-	-	-	0	0	0
300	-	-	-	-	0	0	0
400	-	-	-	-	0	0	0
500	-	-	-	-	0	0	0
600	-	-	-	-	0	0.8	0
700	-	-	-	-	0	1	0
800	-	-	-	-	0	1.43	0
900	-	-	-	-	0.01	2	0.01
1000	-	-	-	-	1	3	1
1200	-	-	-	-	1	4.93	1
1400	-	-	-	-	2.42	7.37	2.48
1600	-	-	-	-	4.09	12.13	4.17
1800	-	-	-	-	5.78	16	6
2000	-	-	-	-	8.15	22.77	8.17
2250	-	-	-	-	11.62	32.32	11.87
2500	-	-	-	-	16.09	44.27	16.09
2750	-	-	-	-	21.45	59.38	21.45
3000	-	-	-	-	28.48	78.93	28.5

Таблица 3.26: Среднее значение $\overline{d}_{\mathbf{NS_2}}(n)$ по выборке случайной величины $d_{\mathbf{NS_2}}(n) = \delta_{\mathbf{NS_2}}(n)/\delta_{\mathbf{NSLS_2}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	1.118229	1.105725	1.299784
200	1.098365	1.082158	1.347887
300	1.087233	1.070273	1.371882
400	1.079495	1.062084	1.383621
500	1.075266	1.056468	1.393316
600	1.071234	1.051952	1.398646
700	1.068637	1.048553	1.403825
800	1.066778	1.045552	1.406457
900	1.064494	1.043317	1.409806
1000	1.063259	1.041272	1.412031
1200	1.060563	1.037912	1.416373
1400	1.058824	1.035336	1.419488
1600	1.057341	1.033218	1.421675
1800	1.055953	1.031370	1.423537
2000	1.055190	1.029865	1.425205
2250	1.053960	1.028205	1.426765
2500	1.053073	1.026847	1.428563
2750	1.052433	1.025678	1.429659
3000	1.051580	1.024646	1.430756

Таблица 3.27: Стандартное отклонение $\overline{\sigma}_{\mathbf{NS_2}}(n)$ по выборке случайной величины $d_{\mathbf{NS_2}}(n) = \delta_{\mathbf{NS_2}}(n)/\delta_{\mathbf{N1LS_2}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	1.118229	1.105725	1.299784
200	1.098365	1.082158	1.347887
300	1.087233	1.070273	1.371882
400	1.079495	1.062084	1.383621
500	1.075266	1.056468	1.393316
600	1.071234	1.051952	1.398646
700	1.068637	1.048553	1.403825
800	1.066778	1.045552	1.406457
900	1.064494	1.043317	1.409806
1000	1.063259	1.041272	1.412031
1200	1.060563	1.037912	1.416373
1400	1.058824	1.035336	1.419488
1600	1.057341	1.033218	1.421675
1800	1.055953	1.031370	1.423537
2000	1.055190	1.029865	1.425205
2250	1.053960	1.028205	1.426765
2500	1.053073	1.026847	1.428563
2750	1.052433	1.025678	1.429659
3000	1.051580	1.024646	1.430756

Таблица 3.28: Границы доверительного интервала для случайной величины $d_{\mathbf{NS_2}}(n) = \delta_{\mathbf{NS_2}}(n)/\delta_{\mathbf{NLS_2}}(n)$ при уровне значимости 0.05.

$n \setminus p$	0.33	0.5	0.67
100	[1.116443, 1.120014]	[1.104100, 1.107349]	[1.295245, 1.304322]
200	[1.097413, 1.099317]	[1.081527, 1.082789]	[1.344647, 1.351128]
300	[1.086544, 1.087923]	[1.069779, 1.070768]	[1.369551, 1.374214]
400	[1.078893, 1.080098]	[1.061764, 1.062403]	[1.381628, 1.385613]
500	[1.074676, 1.075856]	[1.056201, 1.056734]	[1.391876, 1.394756]
600	[1.070696, 1.071771]	[1.051776, 1.052128]	[1.397062, 1.400229]
700	[1.068122, 1.069153]	[1.048383, 1.048724]	[1.402694, 1.404957]
800	[1.066315, 1.067241]	[1.045385, 1.045719]	[1.405227, 1.407686]
900	[1.064078, 1.064910]	[1.043177, 1.043457]	[1.408879, 1.410733]
1000	[1.062841, 1.063677]	[1.041165, 1.041380]	[1.411142, 1.412921]
1200	[1.060195, 1.060930]	[1.037812, 1.038012]	[1.415465, 1.417280]
1400	[1.058538, 1.059111]	[1.035255, 1.035418]	[1.418731, 1.420245]
1600	[1.057010, 1.057671]	[1.033149, 1.033286]	[1.421002, 1.422349]
1800	[1.055643, 1.056264]	[1.031307, 1.031434]	[1.422915, 1.424159]
2000	[1.054950, 1.055430]	[1.029809, 1.029921]	[1.424609, 1.425802]
2250	[1.053725, 1.054194]	[1.028151, 1.028259]	[1.426209, 1.427321]
2500	[1.052839, 1.053306]	[1.026804, 1.026890]	[1.428020, 1.429106]
2750	[1.052201, 1.052666]	[1.025639, 1.025717]	[1.429130, 1.430188]
3000	[1.051367, 1.051793]	[1.024614, 1.024679]	[1.430321, 1.431192]

Таблица 3.29: Статистика Колмогорова-Смирнова для эмпирической функции распределения случайной величины $d_{\mathbf{NS_2}}(n) = \delta_{\mathbf{NS_2}}(n)/\delta_{\mathbf{NLS_2}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.47	0.62	0.48
200	0.63	1.02	0.5
300	0.97	1.1	1.06
400	0.65	0.81	0.55
500	0.55	0.7	0.57
600	0.83	0.37	0.74
700	0.59	0.74	0.65
800	0.89	0.74	1.19
900	0.83	0.96	0.49
1000	0.89	0.47	0.49
1200	0.99	0.42	1.06
1400	0.91	0.53	1.11
1600	0.95	0.84	1.13
1800	0.91	0.75	0.77
2000	1.05	0.47	0.41
2250	0.58	0.63	0.77
2500	0.82	0.95	1.14
2750	0.84	0.84	0.62
3000	0.82	0.51	0.61

Таблица 3.30: Среднее значение $\overline{d}_{\mathbf{PNSLS_2}}(n)$ по выборке случайной величины $d_{\mathbf{PNSLS_2}}(n) = \delta_{\mathbf{PNLSS_2}}(n)/\delta_{\mathbf{NSLS_2}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	1.010667	1.011313	1
200	1.007131	1.007537	1
300	1.005559	1.005720	1
400	1.004233	1.004467	1
500	1.003703	1.003717	1
600	1.003163	1.003386	1
700	1.002953	1.003066	1
800	1.002725	1.002648	1
900	1.002383	1.002516	1
1000	1.002172	1.002265	1
1200	1.001813	1.001876	1
1400	1.001552	1.001714	1
1600	1.001421	1.001490	1
1800	1.001402	1.001328	1
2000	1.001240	1.001154	1
2250	1.001039	1.001076	1
2500	1.000981	1.001007	1
2750	1.000920	1.000939	1
3000	1.000822	1.000885	1

Таблица 3.31: Стандартное отклонение $\overline{\sigma}_{\mathbf{PNSLS_2}}(n)$ по выборке случайной величины $d_{\mathbf{PNSLS_2}}(n) = \delta_{\mathbf{PNSLS_2}}(n)/\delta_{\mathbf{N1LS_2}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.006126	0.006854	0
200	0.003385	0.003101	0
300	0.002079	0.002193	0
400	0.002011	0.001922	0
500	0.001251	0.001420	0
600	0.001213	0.001154	0
700	0.001104	0.001122	0
800	0.000916	0.000927	0
900	0.000772	0.000905	0
1000	0.000671	0.000710	0
1200	0.000598	0.000671	0
1400	0.000525	0.000532	0
1600	0.000478	0.000474	0
1800	0.000422	0.000429	0
2000	0.000368	0.000399	0
2250	0.000308	0.000327	0
2500	0.000284	0.000287	0
2750	0.000282	0.000290	0
3000	0.000227	0.000262	0

Таблица 3.32: Границы доверительного интервала для случайной величины $d_{\mathbf{PNSLS_2}}(n) = \delta_{\mathbf{PNSLS_2}}(n)/\delta_{\mathbf{NLS_2}}(n)$ при уровне значимости 0.05.

$n \setminus p$	0.33	0.5	0.67
100	[1.009466, 1.011867]	[1.009969, 1.012656]	[1, 1]
200	[1.006468, 1.007795]	[1.006929, 1.008145]	[1, 1]
300	[1.005151, 1.005966]	[1.005290, 1.006150]	[1, 1]
400	[1.003839, 1.004627]	[1.004090, 1.004844]	[1, 1]
500	[1.003458, 1.003948]	[1.003439, 1.003995]	[1, 1]
600	[1.002925, 1.003401]	[1.003160, 1.003613]	[1, 1]
700	[1.002736, 1.003169]	[1.002846, 1.003286]	[1, 1]
800	[1.002545, 1.002904]	[1.002466, 1.002829]	[1, 1]
900	[1.002232, 1.002535]	[1.002338, 1.002693]	[1, 1]
1000	[1.002040, 1.002303]	[1.002126, 1.002405]	[1, 1]
1200	[1.001696, 1.001931]	[1.001744, 1.002007]	[1, 1]
1400	[1.001449, 1.001655]	[1.001609, 1.001818]	[1, 1]
1600	[1.001327, 1.001515]	[1.001397, 1.001583]	[1, 1]
1800	[1.001319, 1.001484]	[1.001244, 1.001412]	[1, 1]
2000	[1.001168, 1.001313]	[1.001076, 1.001232]	[1, 1]
2250	[1.000979, 1.001100]	[1.001012, 1.001141]	[1, 1]
2500	[1.000926, 1.001037]	[1.000951, 1.001063]	[1, 1]
2750	[1.000865, 1.000976]	[1.000882, 1.000996]	[1, 1]
3000	[1.000777, 1.000866]	[1.000834, 1.000936]	[1, 1]

Таблица 3.33: Статистика Колмогорова-Смирнова для эмпирической функции распределения случайной величины $d_{\mathbf{PNSLS_2}}(n) = \delta_{\mathbf{PNSLS_2}}(n)/\delta_{\mathbf{NLS_2}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.68	0.81	-
200	0.52	1.05	-
300	0.6	0.79	-
400	0.51	0.64	-
500	0.68	0.74	-
600	0.5	0.65	-
700	0.63	0.51	-
800	0.46	0.62	-
900	0.89	0.55	-
1000	0.51	0.83	-
1200	0.44	0.87	-
1400	0.74	1.01	-
1600	0.49	0.72	-
1800	0.74	0.55	-
2000	0.55	0.67	-
2250	0.51	0.62	-
2500	0.65	0.67	-
2750	0.53	0.49	-
3000	0.52	0.62	-

Данные экспериментального исследования алгоритмов для задачи $\mathbf{GC_{\leqslant 3}}$

Таблица 3.34: Среднее время работы алгоритмов в секундах при $p = 0.33$

n	**BF**	**BBM**	**CBC**	**Gurobi**	$\mathbf{NLS_{\leqslant 3}}$	$\mathbf{DN_{\leqslant 3}}$	$\mathbf{ND1LS_{\leqslant 3}}$
15	1	0	1000.15	7.7	0	0	0
20	493.85	0	-	217.76	0	0	0
25	-	0.09	-	1500.68	0	0	0
30	-	11.57	-	-	0	0	0
35	-	336.51	-	-	0	0	0
100	-	-	-	-	0.04	0	0
150	-	-	-	-	2.62	0	0
200	-	-	-	-	6.42	0.09	0.2
250	-	-	-	-	12.04	2	2
300	-	-	-	-	21.12	4	4
350	-	-	-	-	31.4	7.57	7.98
400	-	-	-	-	50.32	13.28	13.49
500	-	-	-	-	86.29	31.57	31.8

Таблица 3.35: Среднее время работы алгоритмов в секундах при $p = 0.5$

n	**BF**	**BBM**	**CBC**	**Gurobi**	$\mathbf{NLS_{\leqslant 3}}$	$\mathbf{DN_{\leqslant 3}}$	$\mathbf{ND1LS_{\leqslant 3}}$
15	1	0	1218.6	10.54	0	0	0
20	493.85	0	-	350.69	0	0	0
25	-	1.29	-	2324.65	0	0	0
30	-	67.31	-	-	0	0	0
35	-	2402.47	-	-	0	0	0
100	-	-	-	-	0	0	0
150	-	-	-	-	1	0	0
200	-	-	-	-	3	0.75	1
250	-	-	-	-	5.97	2	2
300	-	-	-	-	10.61	4.17	4.36
350	-	-	-	-	15.91	8.97	9.08
400	-	-	-	-	25.77	15.3	15.41
500	-	-	-	-	44.72	35.5	36.41

Таблица 3.36: Среднее время работы алгоритмов в секундах при $p = 0.67$

n	**BF**	**BBM**	**CBC**	**Gurobi**	**NLS$_{\leqslant 3}$**	**DN$_{\leqslant 3}$**	**ND1LS$_{\leqslant 3}$**
15	1	0	717.42	3.3	0	0	0
20	493.85	0	-	200	0	0	0
25	-	0.05	-	1026.4	0	0	0
30	-	3.39	-	-	0	0	0
35	-	38.36	-	-	0	0	0
100	-	-	-	-	0	0	0
150	-	-	-	-	0	0	0
200	-	-	-	-	1	0.1	0.17
250	-	-	-	-	2.35	2	2
300	-	-	-	-	4.15	4	4
350	-	-	-	-	6.15	7.84	7.77
400	-	-	-	-	9.98	13.15	13.09
500	-	-	-	-	15	30.5	27.5

Таблица 3.37: Среднее значение $\overline{d}_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n)$ по выборке случайной величины $d_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n) = \delta_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n)/\delta_{\mathbf{DN1LS}_{\leqslant \mathbf{3}}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	1.049456	1.068099	1.315502
150	1.045360	1.060758	1.343159
200	1.041924	1.056990	1.360641
250	1.038877	1.051277	1.370805
300	1.037161	1.048811	1.379731
350	1.035233	1.045934	1.386063
400	1.034179	1.044121	1.389215
500	1.031513	1.040499	1.401489

Таблица 3.38: Стандартное отклонение $\overline{\sigma}_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n)$ по выборке случайной величины $d_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n) = \delta_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n)/\delta_{\mathbf{DN1LS}_{\leqslant \mathbf{3}}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.010769	0.010888	0.026728
150	0.006777	0.007639	0.020925
200	0.005077	0.005420	0.015902
250	0.003852	0.004291	0.013242
300	0.003810	0.003174	0.012157
350	0.003268	0.003027	0.009905
400	0.002664	0.002754	0.009258
500	0.001968	0.002257	0.009542

Таблица 3.39: Границы доверительного интервала для случайной величины $d_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n) = \delta_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n)/\delta_{\mathbf{DN1LS}_{\leqslant \mathbf{3}}}(n)$ при уровне значимости 0.05.

$n \setminus p$	0.33	0.5	0.67
100	[1.047345, 1.051567]	[1.065965, 1.070233]	[1.310264, 1.320741]
150	[1.044032, 1.046689]	[1.059261, 1.062256]	[1.339058, 1.347261]
200	[1.040929, 1.042919]	[1.055927, 1.058052]	[1.357524, 1.363758]
250	[1.038122, 1.039632]	[1.050436, 1.052118]	[1.368210, 1.373401]
300	[1.036415, 1.037908]	[1.048189, 1.049433]	[1.377348, 1.382114]
350	[1.034593, 1.035874]	[1.045340, 1.046527]	[1.384122, 1.388005]
400	[1.033657, 1.034701]	[1.043581, 1.044660]	[1.387400, 1.391030]
500	[1.031127, 1.031899]	[1.040057, 1.040942]	[1.399619, 1.403359]

Таблица 3.40: Статистика Колмогорова-Смирнова для эмпирической функции распределения случайной величины $d_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n) = \delta_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n)/\delta_{\mathbf{NLS}_{\leqslant \mathbf{3}}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.79	0.71	0.59
150	0.73	0.53	0.81
200	0.54	0.54	0.71
250	0.47	0.62	0.93
300	0.61	0.59	0.48
350	0.72	1.15	0.58
400	0.64	0.72	0.5
500	0.65	0.47	0.42

Таблица 3.41: Среднее значение $\overline{d}_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n)$ по выборке случайной величины $d_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n) = \delta_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n)/\delta_{\mathbf{DN1LS}_{\leqslant \mathbf{3}}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	1.113851	1.100268	1.321948
150	1.104028	1.090561	1.353711
200	1.096201	1.083938	1.374634
250	1.089690	1.076159	1.387213
300	1.085626	1.072138	1.398479
350	1.081453	1.067913	1.406267
400	1.078163	1.065272	1.410715
500	1.072602	1.059931	1.424582

Таблица 3.42: Стандартное отклонение $\overline{\sigma}_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n)$ по выборке случайной величины $d_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n) = \delta_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n)/\delta_{\mathbf{DN1LS}_{\leqslant \mathbf{3}}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.012171	0.011030	0.027791
150	0.007594	0.008327	0.022851
200	0.005731	0.004961	0.017765
250	0.004921	0.004477	0.014763
300	0.003941	0.003456	0.013324
350	0.003349	0.002917	0.011189
400	0.003164	0.002653	0.010653
500	0.002237	0.002448	0.011586

Таблица 3.43: Границы доверительного интервала для случайной величины $d_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n) = \delta_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n)/\delta_{\mathbf{DN1LS}_{\leqslant \mathbf{3}}}(n)$ при уровне значимости 0.05.

$n \setminus p$	0.33	0.5	0.67
100	[1.111465, 1.116237]	[1.098106, 1.102429]	[1.316500, 1.327395]
150	[1.102540, 1.105517]	[1.088929, 1.092194]	[1.349232, 1.358190]
200	[1.095077, 1.097324]	[1.082966, 1.084910]	[1.371152, 1.378116]
250	[1.088726, 1.090655]	[1.075281, 1.077036]	[1.384319, 1.390107]
300	[1.084853, 1.086398]	[1.071461, 1.072816]	[1.395867, 1.401091]
350	[1.080796, 1.082109]	[1.067341, 1.068484]	[1.404074, 1.408460]
400	[1.077542, 1.078783]	[1.064752, 1.065792]	[1.408627, 1.412804]
500	[1.072164, 1.073041]	[1.059452, 1.060411]	[1.422311, 1.426853]

Таблица 3.44: Статистика Колмогорова-Смирнова для эмпирической функции распределения случайной величины $d_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n) = \delta_{\mathbf{DN}_{\leqslant \mathbf{3}}}(n)/\delta_{\mathbf{DN1LS}_{\leqslant \mathbf{3}}}(n)$.

$n \setminus p$	0.33	0.5	0.67
100	0.6	0.75	0.52
150	0.7	0.5	0.65
200	0.33	0.58	0.67
250	0.71	0.78	0.88
300	0.44	0.86	0.71
350	0.39	0.86	0.97
400	1	0.66	0.74
500	0.86	0.65	0.8