Московский государственный университет имени М.В.Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра математических методов прогнозирования

АЛЕШИН Илья Сергеевич

Обучаемые процедуры выделения плотных подматриц в разреженных матрицах

ДИПЛОМНАЯ РАБОТА

Научный руководитель:

профессор, член-корр. РАН

К.В.Рудаков

Москва, 2015

Содержание

1 Введение

Будем рассматривать произвольные разреженные матрицы **(sparse matrix)**. Разреженная матрица определяется как матрица с преимущественно нулевыми элементами. Среди специалистов нет единства в определении того, какое именно количество ненулевых элементов делает матрицу разреженной. Одним из возможных вариантов для $m \times n$ матриц является $o(m * n)$ ненулевых элементов.

Часто встречаются ситуации, когда в исходной матрице ненулевые элементы распределены неравномерно. Например, возможны случаи, когда существуют подмножества строк и столбцов исходной матрицы, содержащие значительную долю ненулевых элементов. Назовём подматрицы, образованные такими подмножествами, **плотными** (формального определения плотности мы давать не будем, позже убедимся в том, что само понятие плотности плохо формализовано). На данном этапе ограничимся наиболее общим и интуитивно простым понятием плотности, определив её как отношение числа ненулевых элементов матрицы к её размеру.

Задача выделения плотных подматриц имеет большое прикладное значение.

В частности, во многих задачах анализа клиентских сред и коллаборативной фильтрации, данные хранятся в виде матрицы **«клиент-сервис»** («пользователь-предмет», «субъект-объект»), где в позиции (i, j) этой матрицы стоит некоторая характеристика использования i-ым клиентом j-ого сервиса. Например, клиентами могут служить пользователи, просматривающие фильмы, сервисами — сами фильмы, (i, j)-элемент представляет собой оценку от 1 до 5, которую i-ый пользователь поставил j-ому фильму. Нулевые элементы соответствуют тому, что пользователь не смотрел данный фильм или не оценил его (одной из таких задач является известная задача Netflix).

Примерами задач могут служить:

- задачи тематического моделирования (кластеризовать данные по темам, по объекту/субъекту найти близкие)
- задачи анализа социальной сети (поиск единомышленников (like-minded people))
- задачи персонализации предложений (выделить интересы клиентов, сегментировать клиентскую базу, выдать клиенту рекомендации)

и многие другие. Немалая часть задач характеризуется сильной разреженностью данных. Большинство пользователей используют очень малую долю всех сервисов. В век информационного взрыва количество объектов даже в одной категории (такой, как фильмы, музыка, книги, новости, веб-сайты) стало настолько большим, что отдельный человек не способен просмотреть даже их значительную часть. Становится необходимым находить группы схожих клиентов и схожих сервисов, чтобы строить прогнозы и делать рекомендательные выводы.

2 Постановка задачи

В качестве исходной конфигурации будем рассматривать произвольную разреженную матрицу $A^{m \times n} = \|a_{ij}\|$ (вообще говоря, m не равно n), элементы которой представляют собой элементы множества $\{0, 1\}$.

Обозначим $U = \{u_i\}_{i=1}^{m}$ — множество строк матрицы A, $R = \{r_j\}_{j=1}^{n}$ — множество столбцов матрицы A.

Если интерпретировать матрицу A как матрицу «субъекты-объекты», то множество U будет представлять собой множество субъектов, множество R — множество объектов.

Как правило, ненулевые элементы не образуют чёткой структуры и представляют собой случайно разбросанные точки. Однако, возможно существуют такие перестановки строк

$\{u_{i_1}u_{i_2}\dots u_{i_m}\}$ и столбцов $\{r_{j_1}r_{j_2}\dots r_{j_n}\}$ исходной матрицы, что подмножества единичных элементов будут образовывать сгущения некоторой формы. Назовём эквивалентными преобразованиями матрицы подстановки местами её строк или столбцов.

Цель данной работы состоит в разработке и исследовании методов, которые эквивалентными преобразованиями перемещают единичные элементы так, чтобы сформировать некоторые конфигурации из единичных элементов, которые будут образовывать «плотные» подматрицы. В интерпретации «клиенты-сервисы» это означает поиск похожих клиентов в смысле использования одинаковых сервисов и поиск схожих сервисов в смысле использования одинаковыми клиентами.

Другими словами, решаемая задача заключается в поиске подмножетсв строк $\{i_{p1},\dots,i_{pr_p}\}$, $\{j_{p1},\dots,j_{pq_p}\}$, где $p=1,\dots,P$, P — число сгущений. Число P может быть известно из условий задачи или определено алгоритмами, которые будут предложены в дальнейших разделах.

3 Генерация модельных данных

Для проведения экспериментов был написан генератор модельных данных, работа которого заключается в следующем: генерируются случайные разреженные матрицы со сгущениями, размеры которых m и n лежат в некотором диапазоне.

Рис. 1: Пример сгенерированной разреженной матрицы с двумя сгущениями после третьего и четвёртого этапов (матрицы A и B)

Генерация осуществляется следующей последовательностью действий:

- сгенерировать матрицу с равномерно распределёнными ненулевыми элементами (используется параметр, отвечающий за отношения числа ненулевых элементов к размеру матрицы, которой выбирается случайно из некоторого диапазона, чтобы матрица была разреженной)
- выбрать несколько случайных подматриц (в зависимости от того, сколько сгущений необходимо сгенерировать), размеры которых лежат в некотором диапазоне; при этом порядок размеров подматриц меньше порядка матрицы, например $O(\sqrt{n}\,)$ или $O(n^{\alpha}\,)$, где $\alpha \leqslant 1-\varepsilon$, где $\varepsilon \geqslant 0$ — некоторый порог
- заполнить данные подматрицы некоторым количеством единиц; заполнение можно вести по определённому закону с определённой степенью интенсивности заполнения (например, можно использовать алгоритмы дискретизации изображений (Брезенхэма и др.), см. [7]) (обозначим: A — матрица, полученная на данном этапе)
- осуществить большое случайных число транспозиций строк и столбцов (обозначим: B — матрица, полученная на данном этапе)

Пример генерации матрицы представлен на рисунке 1. Элементы, принадлежащие разным сгущениям, визуализированы разными цветами. Шумовые элементы визуализированы зелёным цветом. Размер матрицы 153×112, число ненулевых элементов —233.

4 Алгоритмы выделения сгущений

4.1 «Наивный» алгоритм

Сгенерируем произвольную разреженную матрицу с одним сгущением (размеры в диапазоне от 100 до 500) (рис. 2).

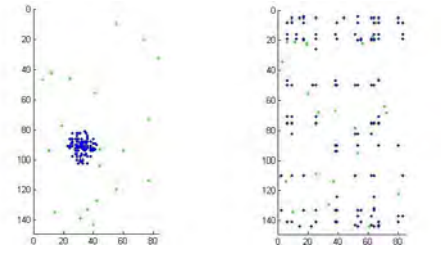

Рис. 2: Матрицы A и B

Далее упорядочим по возрастанию числа ненулевых элементов строки, затем столбцы. Получим следующую конфигурацию (рис. 3)

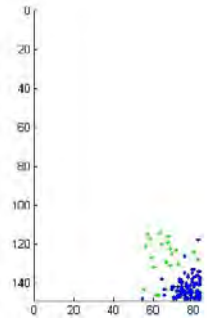

Рис. 3: Упорядоченная матрица

Перенесём данное сгущение в центр матрицы эквивалентными преобразованиями так, чтобы по мере удаления от центра число элементов в строках и столбцах уменьшалось (рис. 4). Обозначим данную матрицу C.

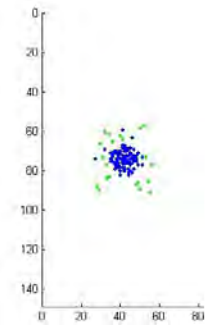

Рис. 4: Матрица C

Как видно из эксперимента, данный алгоритм идеально восстановил сгущение. Однако, проиллюстриурем результаты алгоритма на матрицах, содержащих более одной плотной подматрицы (приводятся только матрицы A и C)

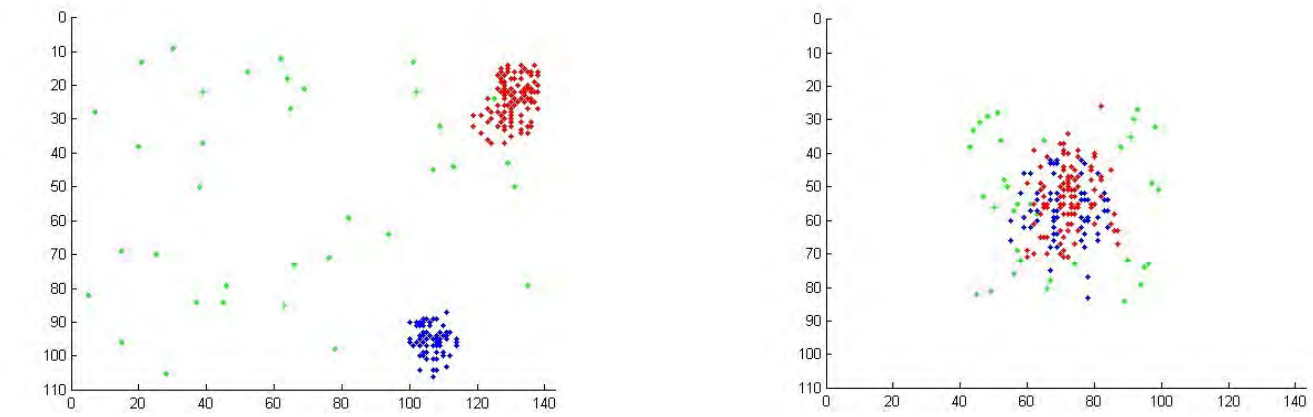

Рис. 5: матрицы A(два сгущения) и C

Можно наблюдать, что алгоритм восстановил плотную подматрицу, однако все сгущения смешались в одно. Если интерпретировать матрицу как «клиенты-сервисы», то получается, что алгоритм разные группы клиентов и сервисов смешал в одну большую группу. Конечно, нам хотелось бы получить подмножества точек, обособленные от других подмножеств, чтобы точки из разных подматриц исходной конфигурации были отделены друг от друга в результирующей конфигурации. Поэтому целью является не выделение каких-либо произвольных плотных подматриц, а восстановление структуры исходной матрицы, откуда можно сделать вывод о том, что результаты данного алгоритма в этом смысле непригодны.

Выводы: данный алгоритм очень прост в реализации, имеет высокую скорость работы ($O(\overline{nlogn})$), очень эффективен в случае одного сгущения, но в случаях большего числа сгущений его результаты неинтерпретируемы в терминах «клиенты-сервисы».

4.2 Процедуры, основанные на мерах сходства

Рассмотрим $U = \{u_i\}_{i=1}^{m}$ — множество строк матрицы A, $R = \{r_j\}_{j=1}^{n}$ — множество столбцов матрицы A.

Введём меру сходства между субъектами и объектами:

$$K(u,v) = \sum_{i=1}^{n} [u_i \& v_i],$$

где $K(u,v)$ — функция сходства между объектами u и v. Также можно ввести меры сходства другим способом

Идея алгоритма заключается в нахождении групп строк и столбцов, средняя оценка K внутри которых больше некоторых порогов для субъектов и объектов. Пороги сходства подаются на вход данного алгоритма.

Алгоритм, основанный на мерах сходства.

Вход: A - исходная матрица, ε — порог сходства внутри групп строк, δ — порог сходства внутри групп столбцов, U — множество номеров строк, V — множество номеров столбцов, K — число допустимых шагов выбора

Выход: $u_1, \ldots, u_k, \ldots$ — наборы схожих строк, $v_1, \ldots, v_k, \ldots$ — наборы схожих столбцов

1: **повторять**
2: найти строку u_{i_1} с наибольшей средней оценкой сходства с остальными или с наибольшей оценкой сходства с некоторой строкой из оставшихся :$i_1 = argmax_{i_1} \frac{1}{|U|} \sum_{i \in U/i_1} K(u_i, u_{i_1})$, $m = max_i K(u_i, u_{i_1})$
3: **если** $m < \varepsilon$ **то**
4: выход из функции;
5: $U_1 = i_1$
6: $k = 0$;
7: **повторять**
8: выбрать строку из U/U_1 с номером i_k
9: посчитать среднее сходство данной строки со строками из $U/U_1 : m = \sum_{i \in U/U_1} K(u_i, u_{i_k})$
10: **если** $m < \varepsilon$ **то**
11: $k = k + 1$;
12: **если** $k >= K$ **то**
13: выход из цикла
14: **иначе**
15: k=0;
16: присоединить строку с номером i_k к U_1, исключить её из U.
17: **если** $k >= K$ **то**
18: вызов функции для столбцов, лежащих в выбранных строках
19: **пока** $k < K$
20: вернуть группы схожих строк и столбцов U_1, V_1, исключить их из матрицы
21: **пока** 1

После первого этапа работы алгоритма при применении его к матрице на рисунке 5 матрица C будет выглядеть следующим образом(рис.6).

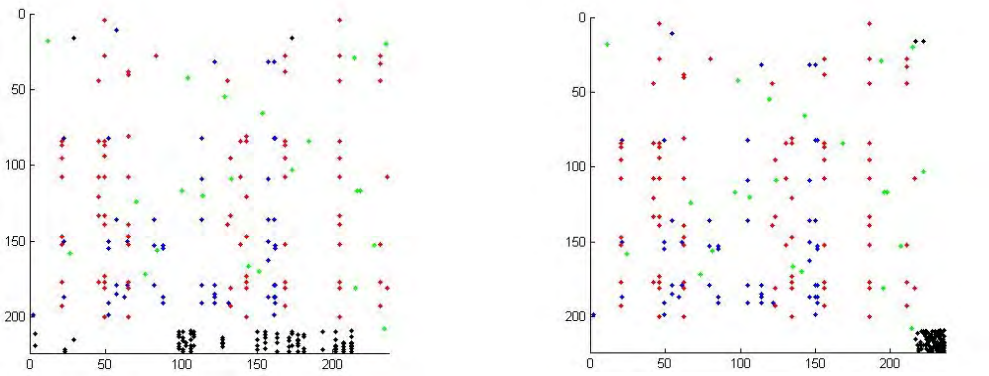

Рис. 6: Результат первого этапа работы алгоритма

Возможные критерии останова:

1. число элементов новой (после исключения строк или столбцов) на каждом этапе матрицы меньше определённого порога n_{min}

2. максимальная мера сходства субъектов u_0 и u_1 на некотором шаге меньше определённого порога α_{min} или число схожих субъектов мало (вероятно, остались строки, элементы

которых представляют собой шум и не принадлежат никаким сгущениям)

Параметр K введён для того, чтобы не осуществлять полный перебор всех строк и столбцов для существенного ускорения скорости работы.

Визуализируем работу данного алгоритма:

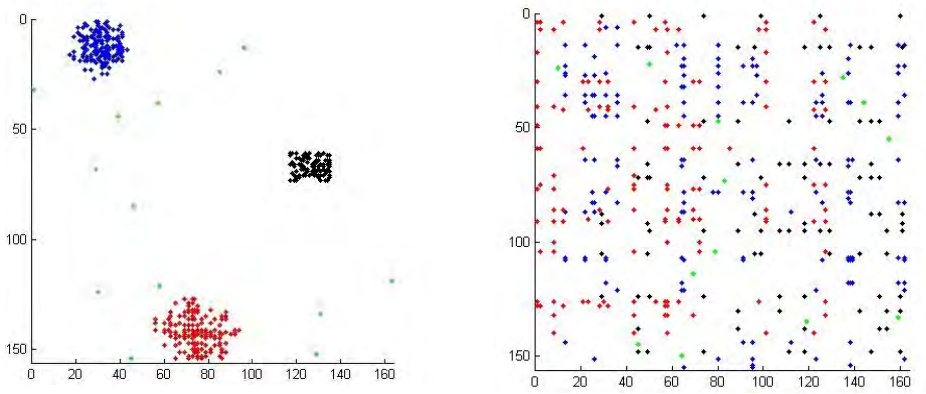

Рис. 7: Матрицы A и B

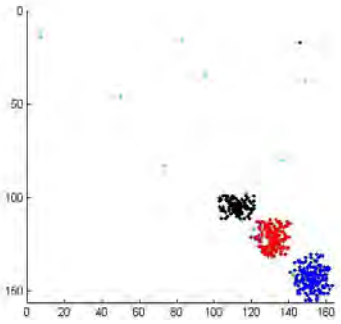

Рис. 8: Матрца C

Как можно видеть, данный алгоритм собрал чётко выраженные, отделённые друг от друга сгущения. Однако, как можно заметить, в рассмотренном случае сгущения не имели пересечений по строкам или по столбцам.

Если рассмотреть случаи, когда в исходной матрицы в некоторых строках или столбцах содержатся элементы из разных сгущений, то в результирующей конфигурации эти сгущения могут «склеиваться»(рис.9)

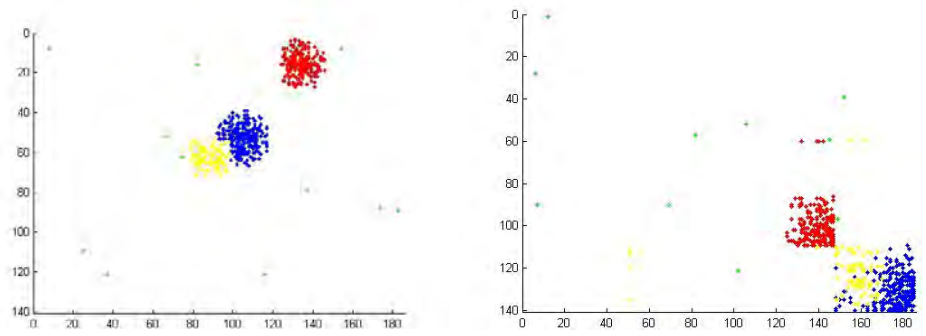

Рис. 9: Матрицы A и C

Если рассматривать матрицу как «клиенты-сервисы», то пересечение по строкам или столбцам означает использование сервисов одним и тем же пользователем или использование пользователями одного и того же сервиса, что и означает сходство сервисов или пользователей, поэтому возможно данные группы не должны быть разделены. Возникает закономерный вопрос вопрос: может ли данный алгоритм разделить такие подматрицы и необходимо ли это осуществить?

Назовём конфигурации эквивалентными, если они могут быть получены друг из друга перестановкой только строк или только столбцов. Очевидно, что если конфигурации эквивалентны, то все элементы разных сгущений, лежащие в одних и тех же строках (столбцах) должны считаться как один кластер точек. Полный ответ на данный вопрос будет дан в секции «Оценка качества решения», когда будут предложены аналитические оценки методов восстановления.

Выводы:

+ данный алгоритм достаточно эффективно восстанавливает исходную конфигурацию
+ метод устойчив к шуму, не зависит от матрицы B, а определяется только начальной конфигурацией
+ легко интерпретируется
− в худшем случае скорость работы $O(n^3 logn)$, что существенно ниже «наивного» алгоритма

4.3 Алгоритмы, основанные на методах глобальной оптимизации

Рассмотрим метод имитации отжига, который относится к методам глобальной оптимизации (см. [2])

Идея данного алгоритма выделения сгущений заключается в том, чтобы собрать ненулевые элементы как можно ближе к главной диагонали.

Введём матрицу весов элементов $W^{m\times n}$

Матрицей Теплица называется диагонально – постоянная матрица.

В общем виде матрица Теплица порядка $n \times n$ имеет вид:

$$\begin{pmatrix} a_0 & a_{-1} & a_{-2} & \dots & \dots & a_{-n+1} \\ a_1 & a_0 & a_{-1} & \ddots & & \vdots \\ a_2 & a_1 & a_0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{-1} & a{-}2 \\ \vdots & & \ddots & a_1 & a_0 & a_{-1} \\ a_{n-1} & \dots & \dots & a_2 & a_1 & a_0 \end{pmatrix}$$

Можно разными способами задать аналог данной матрицы в случае прямоугольной матрицы и таким образом получить матрицу весов W.

Зададим значение элементов таким образом, чтобы они возрастали по мере удаления от главной диагонали (например, линейно, $a_i = k|i|$, или экспоненциально, $a_i = a^{|i|}$, причём положим $a_0 = 0$). Умножим поэлементно матрицу B на матрицу W, просуммируем все элементы текущей матрицы, получим вес текущей конфигурации. Также введём переменную-счётчик k. На каждом шаге будем генерировать случайную перестановку из двух номеров строк или столбцов. При этом возможны следующие варианты:

1. Если вес новой матрицы с переставленными строками или столбцами увеличился или не изменился по сравнению с предыдущей матрицей, то необходимо

- вернуться к старой конфигурации
- увеличить значение счётчика k на единицу
- если при этом значение счётчика выше некоторого порога k_{max} , то, вероятно, старая конфигурация представляет собой точку минимума и лучшего значения веса найти не удастся; при этом алгоритм заканчивает работу

2. Если вес новой матрицы уменьшился, то

   - с некоторой вероятностью (которая зависит от температуры — параметра метода имитации отжига) присвоить текущей матрице новую
   - обнулить счётчик

Очевидно, что вес будет уменьшаться в том случае, если единичные элементы будут располагаться ближе к центральной диагонали матрицы. Возможны некоторые модификации данного алгоритма, основанные на других методах глобальной оптимизации.
На рисунках приведены результаты работы данного алгоритма (визуализированы только исходная и конечная матрицы). Как можно заметить из экспериментов, сгущения получаются более разреженные, чем при работе алгоритмов, основанных на сходстве.

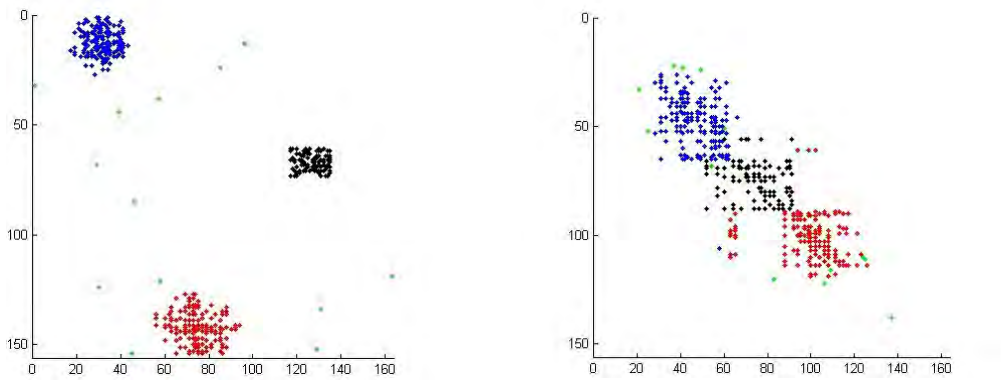

Рис. 10: Матрицы A и C

Выводы:

\+ возможность настраивать множество параметров

\+ высокая скорость работы

– результаты сложно интерпретировать в терминах матрицы «клиенты - сервисы»

– неустойчивость к шуму (результат сильно зависит от матрицы B)

– если матрица сильно отличается от квадратной, сложно построить разумную матрицу весов

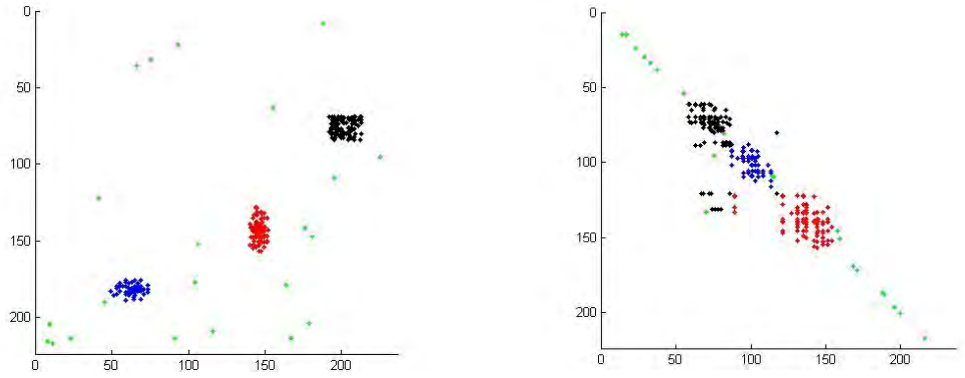

Рис. 11: Пример работы алгоритма (матрицы A и C)

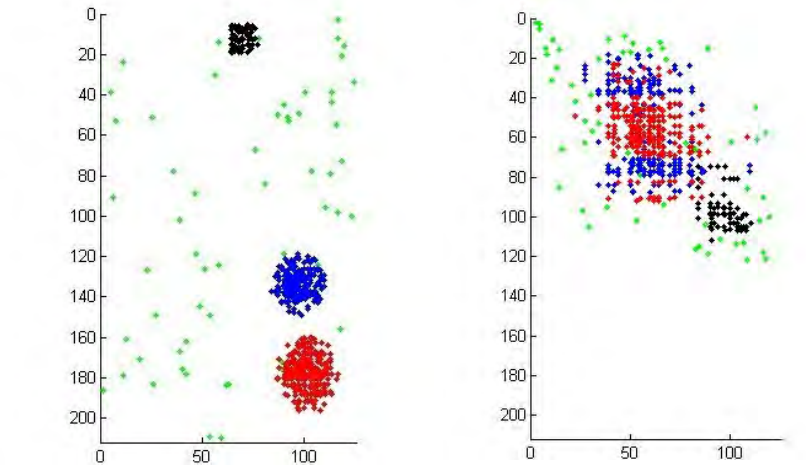

Рис. 12: Пример работы алгоритма (матрицы A и C)

5 Особенности задачи

5.1 NP-трудность задачи

При изучении задачи очень важно знать, к какому классу сложности она принадлежит, поскольку отсюда можно сделать вывод об оценке времени её решения. Покажем, что данная задача принадлежит классу NP-трудных задач ([8]).

Рассмотрим задачу «Клика»:
Вход: граф $G = (V, E)$, число k
Задача: существует ли в G полный подграф с k вершинами?

Также существует вариант данной задачи, когда необходимо найти клику максимального размера.

Покажем, что поиск клики указанного размера k сводится к выделению квадратной полной подматрицы (подматрицы, состоящей целиком из единиц) размера $k \times k$. Заметим, что граф в данной задаче является неориентированным. Построим по исходному графу матрицу A следующим образом: число вершин графа — число строк и столбцов матрицы, a_{ij}=1, если существует ребро, соединяющее i-ую и j-ую вершины графа, $a_{ij} = 0$ в противном случае. Таким образом, в графе G сущесвует полный подграф на k вершинах, если и только если в построенной нами матрице a соответствующие данным вершинам подмножества строк и столбцов образуют полную подматрицу. Таким образом, задача «Клика» сведена к задаче выделения плотных подматриц. Следовательно, данная задача NP-трудна.

5.2 О понятии сгущения

До сих пор мы определяли понятие сгущения интуитивно, основываясь на том, что оно представляет собой плотное подмножество строк и столбцов матрицы. Однако формально границы данной подматрицы никак не определялись.

Определим плотность сгущения как отношение числа ненулевых элементов к его размерам ($\rho = \frac{\#nonZeros}{M*N}$, где M, N — число строк и столбцов в подматрице). Исследуем, как меняется плотность сгущения в зависимости от его размеров.

Предположим, у нас имеется некоторое плотное подмножество строк и столбцов плотности ρ. Будем присоединять строки и столбцы к данному сгущению градиентным алгоритмом (на каждом шаге к сгущению присоединяется строка или столбец, дающие наилучшее изменение среди оставшихся). Графики, иллюстрирующие данные эксперименты, приведены на рисунках 42, 43.

Также данный алгоритм для различных конфигураций запускался из точек (рис. 15).

Как можно видеть, некоторые графики являются строго монотонно убывающими, однако на некоторых можно видеть локальные максимумы. Это происходит, когда $\rho^{t-1} > \rho^t$ и $\rho^t < \rho^{t+1}$, где t — номер итерации. То есть два шага жадного добавления строк и столбцов могут дать больший выигрыш в плотности, чем один шаг.

Для подтверждения данного факта рассмотрим следующий пример.

Пусть у нас имеется матрица 6×6, которая имеет вид:

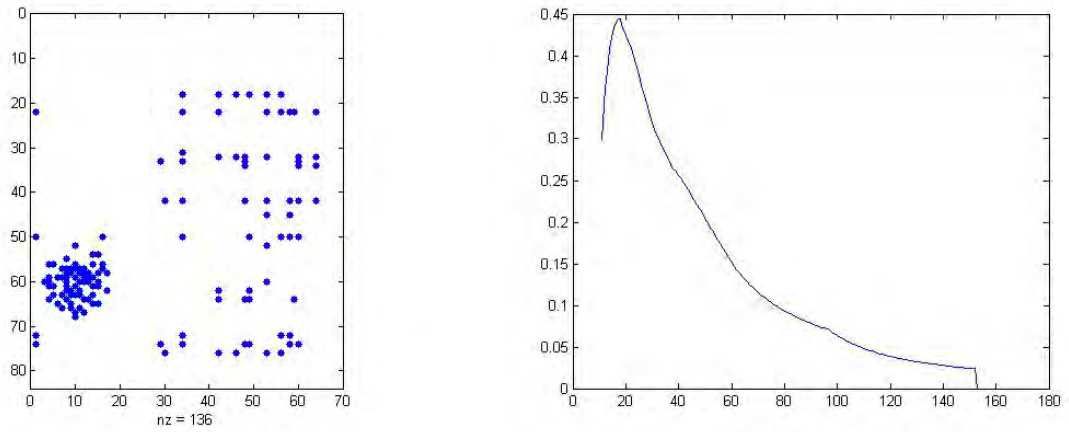

Рис. 13: Конфигурация и график зависимости плотности от числа строк и столбцов в сгущении

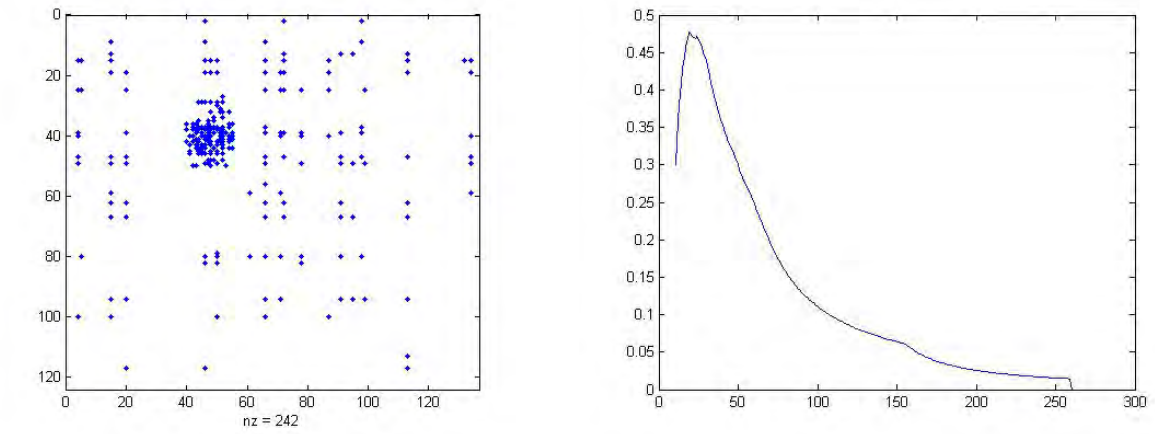

Рис. 14: Конфигурация и график зависимости плотности от числа строк и столбцов в сгущении

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

В качестве начального приближения плотного подмножества выбрана матрица 4×4 в верхнем углу, её плотность равна $\frac{9}{16} = 0.5625$

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Наилучшее изменение плотности даёт присоединение 5-ого столбца или 5-ой строки, при этом $\rho = \frac{11}{20} = 0.55$

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$

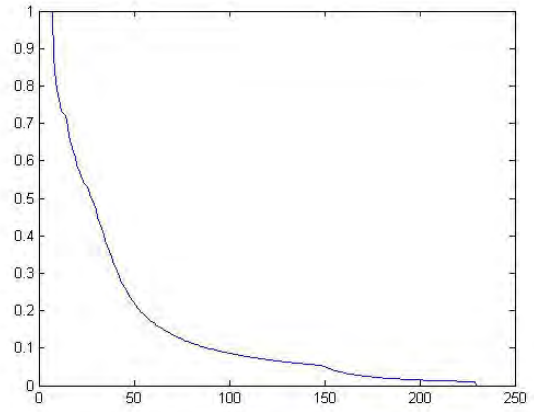
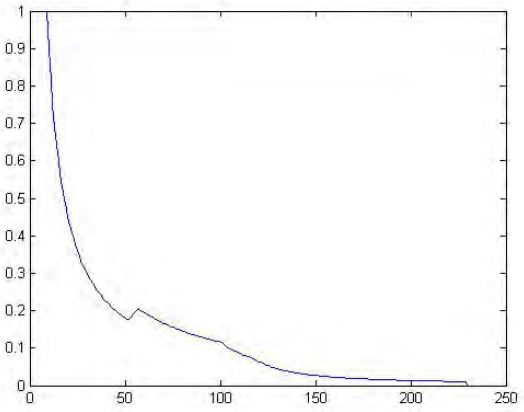

Рис. 15: Графики зависимости плотности от числа строк и столбцов в сгущении (начальная подматрица — точка)

На следующем шаге будет присоединёна 5-ая строка (5-ый столбкец), $\rho = \frac{14}{25} = 0.56 > 0.55$.

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Однако, если использовать жадный алгоритм присоединения только строк или только столбцов, то после достижения глобального максимума плотность монотонно невозрастает.

Исследуем производную функции плотности. Графики зависимости приведены на рис.16, 17.

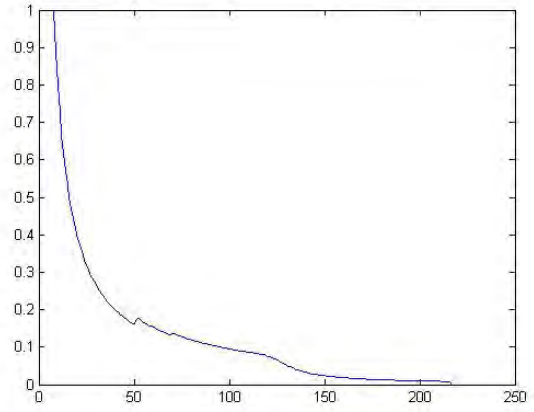
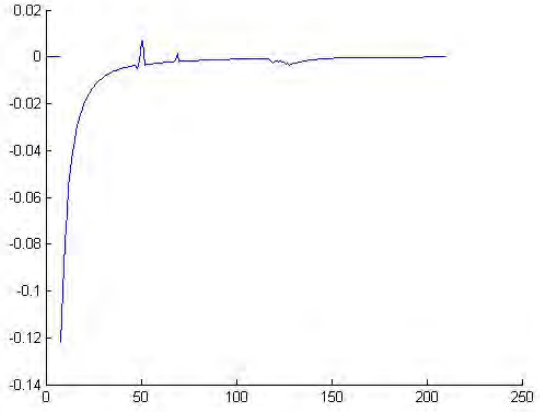

Рис. 16: Графики зависимости плотности и её производной от числа строк и столбцов в сгущении

Как видно из рисунков, в локальных максимумах функции плотности на графике её производной можно наблюдать ярко выраженные пики. Таким образом, можно сделать о том, что в качестве сгущения можно определить плотную подматрицу, на которой достигается локальный максимум зависимости плотности от площади.

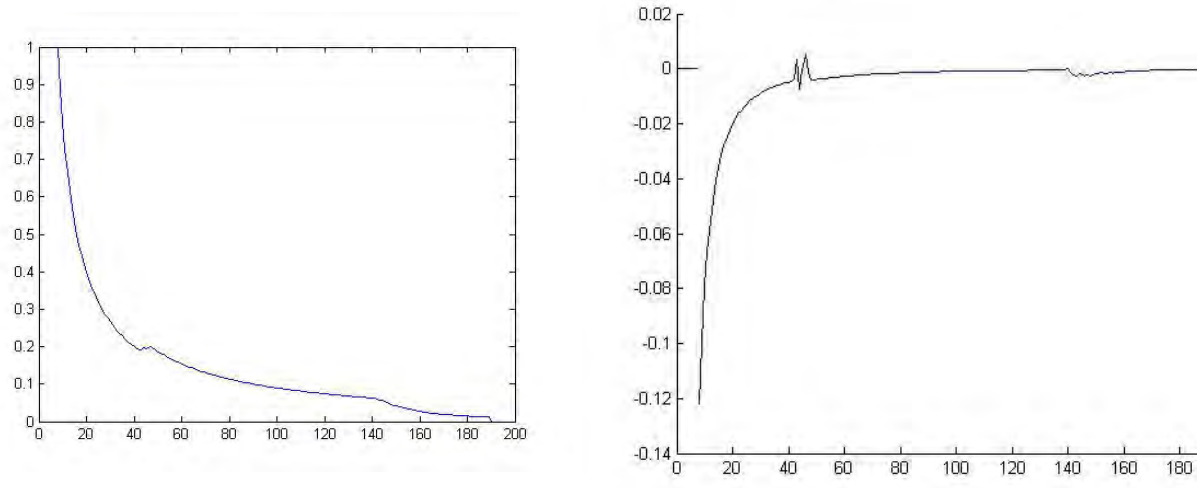

Рис. 17: Графики зависимости плотности и её производной от числа строк и столбцов в сгущении

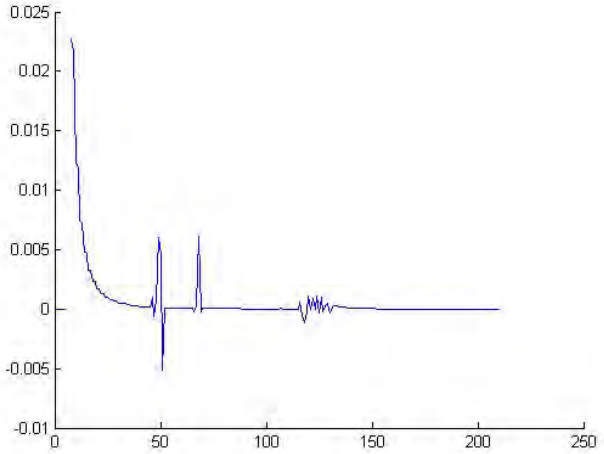

Рис. 18: График зависимости второй производной

6 Оценки качества решений

Исследуя работу алгоритмов, мы оценивали качество решения только визуально. Многие алгоритмы дают очень похожие результаты, поэтому непонятно, какие из них действительно работают лучше. Поэтому необходимо оценивать качество решений аналитически, чем именно мы займёмся в дальнейшей части работы.

6.1 Оценки инцидентности

Применим какой-нибудь из алгоритмов кластеризации к исходной и итоговой конфигурациям (см., например, [1]). Результат работы алгоритмов кластеризации можно представить в виде матрицы инцидентности, которая представляет собой квадратную блочно-диагональную матрицу из нулей и единиц (размер —число точек, соответствующих ненулевым элементам), причём каждый блок соответствует некоторому кластеру. Будем нумеровать кластеры по убыванию числа элементов, чтобы первый блок соответствовал самому большому кластеру и т.д. При этом заметим, что местоположение кластера в матрице не повлияет на вид матрицы инцидентности. Введём обозначения: A_{inc} – матрица инцидентности для исходной конфигурации,

C_{inc} – матрица инцидентности для восстановленной конфигурации.

Определим меру качества решения как (n — число ненулевых элементов матрицы)

$$\sum_{i=1}^{n}\sum_{j=1}^{n}|A_{inc_{ij}} - C_{inc_{ij}}|$$

Очевидно, чем ближе к нулю данная сумма, тем более точным является рассматриваемый алгоритм решения.

Пример построения матрицы инцидентности представлен на рисунках ниже.

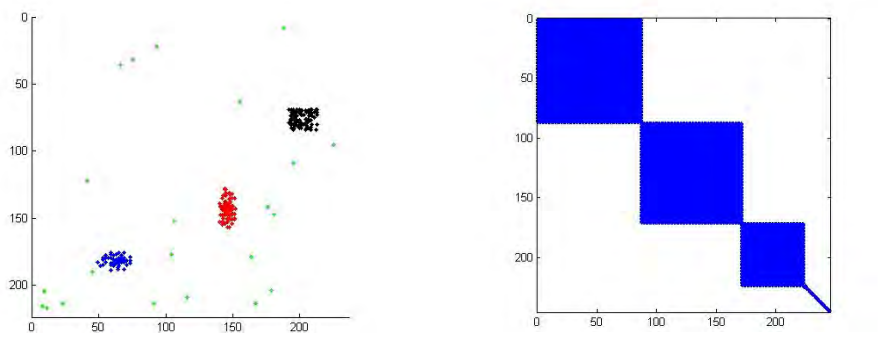

Рис. 19: Матрица A и её матрица инцидентности

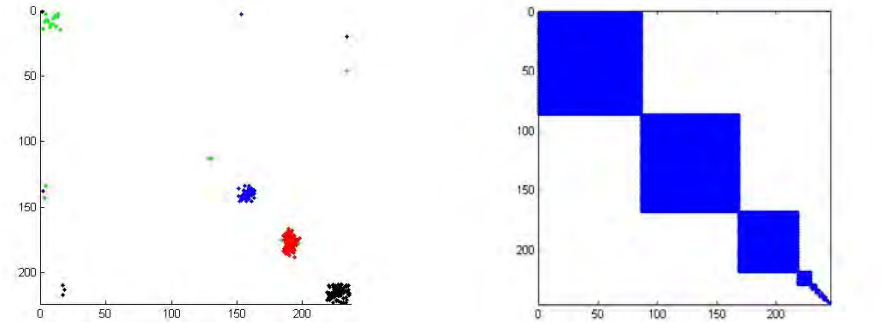

Рис. 20: Алгоритмы, основанные на мере сходства

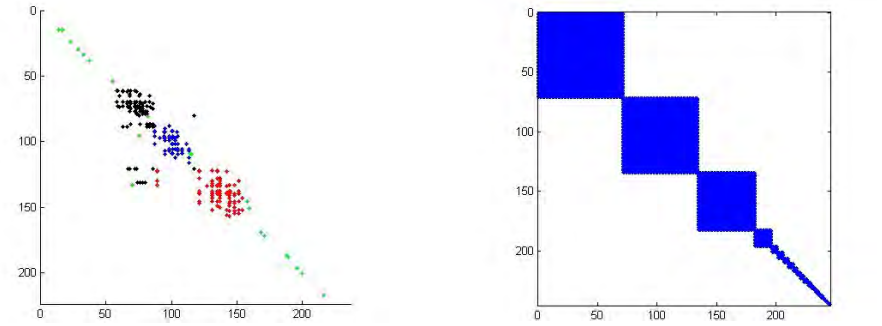

Рис. 21: Алгоритмы, основанные на методах глобальной оптимизации

Анализируя результаты экспериментов, мы можем сделать вывод о том, что алгоритмы, основанные на мере сходства, являются более эффективными, чем методы глобальной оптимизации.

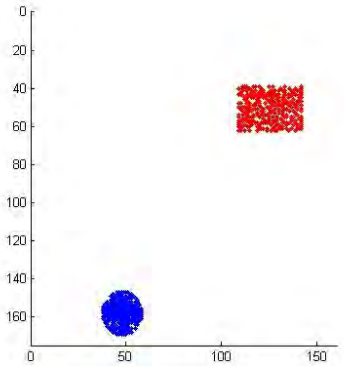

Рис. 22: «Идеальная» конфигурация

Далее попробуем изучить, как влияет уровень шума на качество решения. Рассмотрим следующую конфигурацию (рис.22)

Будем постепенно «зашумлять» данную матрицу (на рисунках ниже показаны некоторые промежуточные результаты)

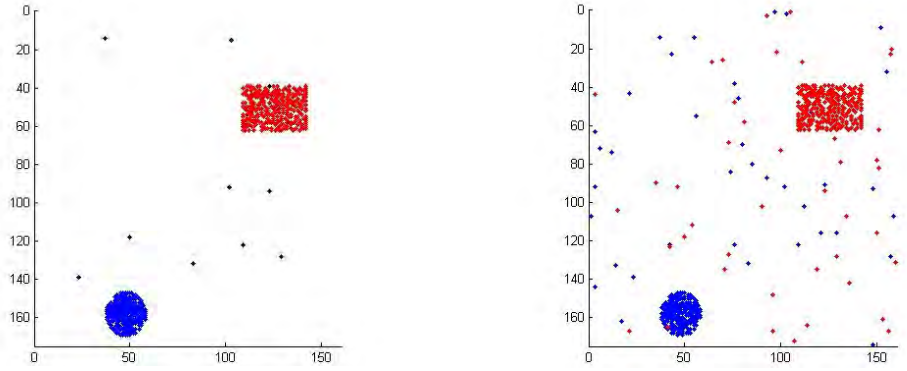

Рис. 23: Матрица с невысоким уровнем шума

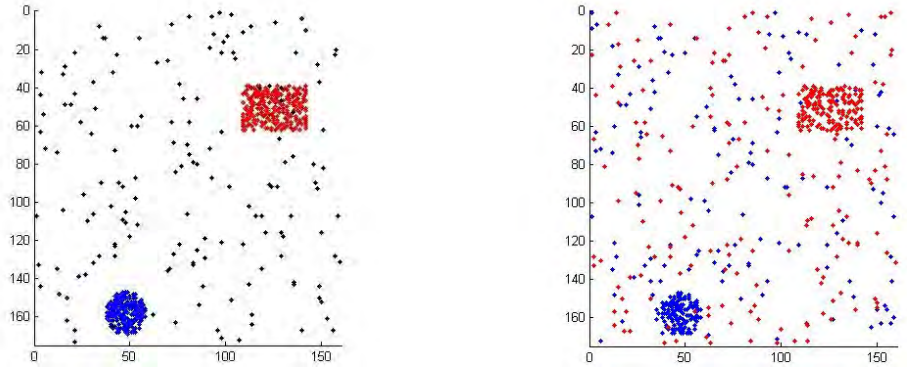

Рис. 24: Сильно «зашумлённая» матрица

Для матрицы на каждом этапе «зашумления» осуществим большое число транспозиций эквивалентными преобразованиями, получим матрицы с переставленными строками и столбцами. Применим алгоритм восстановления сгущений для каждой из таких матриц (рис. 25)

Построим графики зависимости качества решения от уровня шума (по оси oX — доля шума, по oY — введённая нами мера качества, рис. 26, 27). Так как функционал представляет собой расстояние между матрицами, то большим значениям соответствует худшее качество.

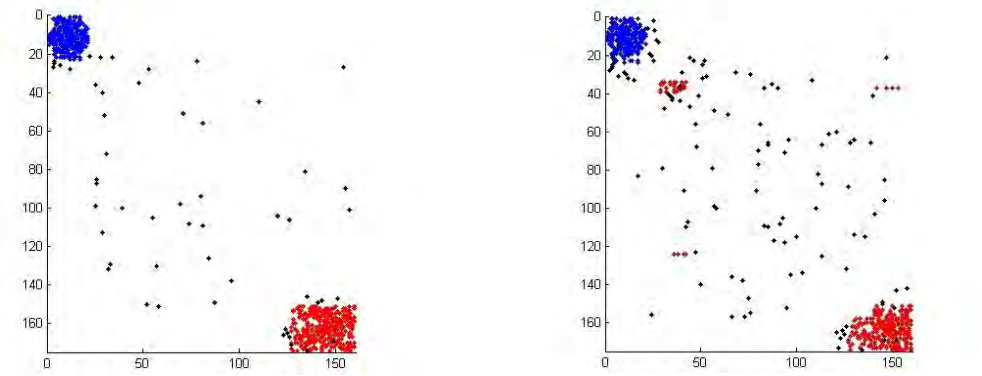

Рис. 25: Результаты восстановления некоторых промежуточных конфигураций

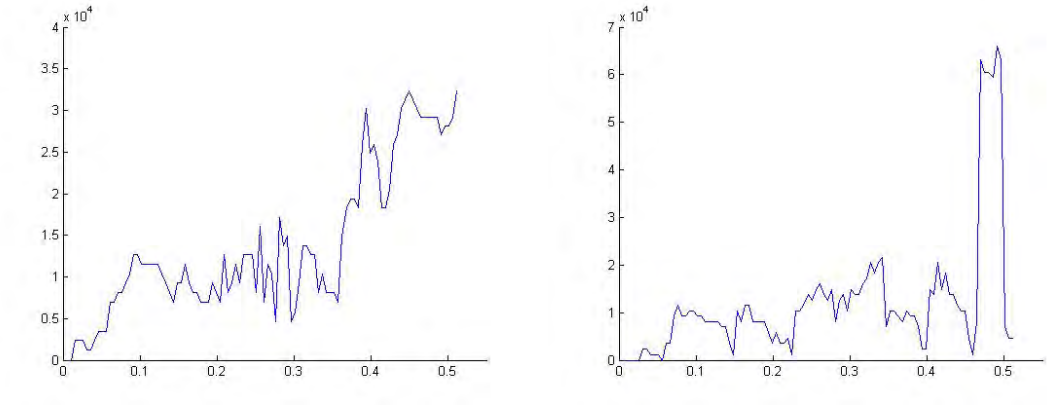

Рис. 26: Результаты экспериментов

Логично было бы предположить, что график должен описывать строго монотонную функцию, однако, как можно видеть, такого не происходит.

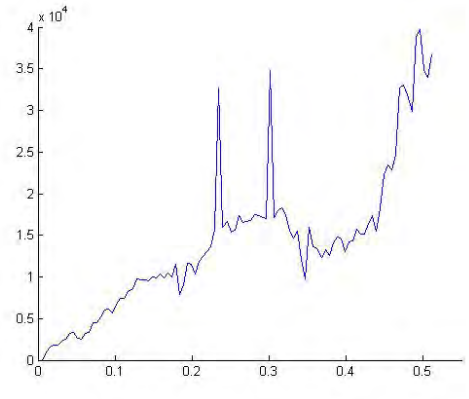

Рис. 27: Усреднение по десяти экспериментам

Попробуем не учитывать шумовые точки при кластеризации исходной и финальной конфигураций, поскольку они могу представлять серьёзную проблему для корректной работы многих

алгоритмов кластеризации (к тому же, неясно, считать ли каждую точку отдельным кластером или объединять их в один).

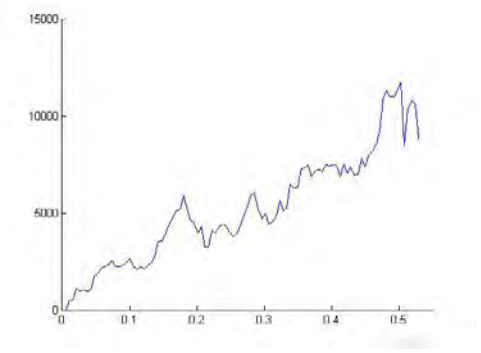

Рис. 28: Усреднение по десяти экспериментам

График приведён на рисунке 28.

Мы видим, что график претерпевает скачки в некоторых точках. Почему же такое происходит? В процессе исследований возникла гипотеза о том, что цена одной ошибки в большом кластере слишком велика, поскольку даже разница в одну точку в исходной и итоговой конфигурациях приводит к значительным изменениям в сумме, задаваемой формулой $\sum_{i=1}^{n}\sum_{j=1}^{n}|A_{inc_{ij}} - C_{inc_{ij}}|$. Также с каждым шагом зашумления размеры кластеров в исходной матрице уменьшаются, поэтому можно преположить, что даже несколько ошибок на более поздних шагах будут имеет меньший вес, чем одна ошибка на предыдущем шаге.

Следовательно, необходимо использовать другую оценку качества решения, более подходящую к данной задаче.

Разделим матрицу инцидентности на каждом шаге на число ненулевых элементов матрицы. Это соответствует использованию формулы $\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|A_{inc_{ij}} - C_{inc_{ij}}|}{n}$. Будем использовать данную меру качества решения(см. 29)

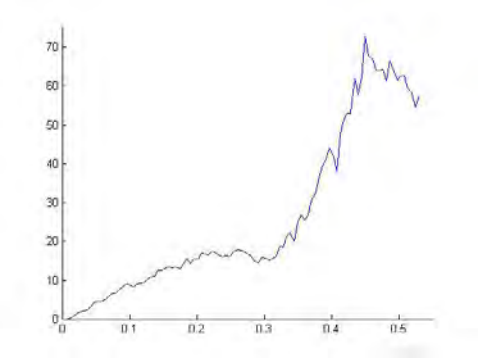

Рис. 29: Усреднение по экспериментам

Как можно видеть, данная мера оказывается более удачной. График претерпевает резкие скачки только при сильном уровне шума.

6.2 Стандартные оценки качества кластеризации

Рассмотрим ненулевые элементы матрицы как элементы линейного векторного пространства X. Каждую точку a матрицы можно описать двумя координатами (a_1, a_2).

Введём следующие функционалы качества:

- Сумма средних внутрикластерных расстояний:

$$F_0 = \sum_{y \in Y} \frac{1}{|K_y|} \sum_{i\,:\,y_i = y} \rho^2(x_i, \mu_y)$$

$K_y = \{x_i \in X^l : y_i = y\}$ —кластер y

μ_y — центр масс кластера y

- Сумма межкластерных расстояний:

$$F_1 = \sum_{y \in Y} \rho^2(\mu_y, \mu)$$

μ — центр масс всей выборки

Будем использовать отношение суммы средних внутрикластерных расстояний исходной и финальной конфигураций как меру качества решения. Межкластерные расстояния использовать не будем, поскольку нам неважно, в каком именно месте матрицы будут собраны кластеры, однако их можно использовать в алгоритмах сбора сгустков, чтобы точки разных сгущений располагались как можно дальше друг от друга.

График зависимости меры качества решения при данном функционале представлен на рис. 30, 31). Большие значения соответствуют лучшему качеству (шумовые точки не учитываются при кластеризации)

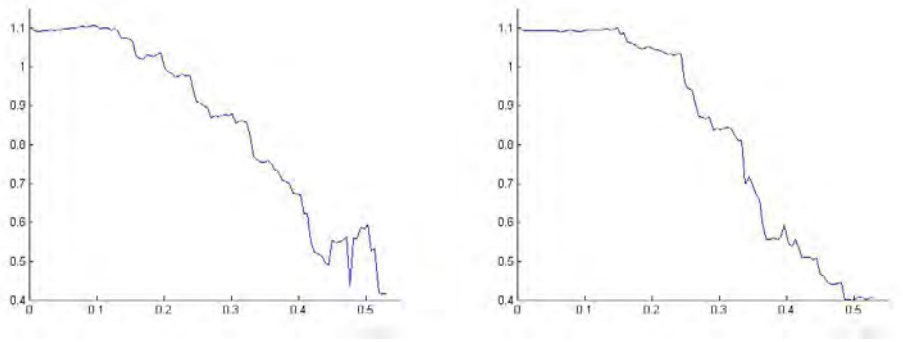

Рис. 30: Результаты экспериментов

Как можно видеть, у данной функции есть большое количество участков монотонности, что оправдывает наши предположения.

Вернёмся к задаче разделения сгущений, имеющих пересечение по строкам или столбцам. Будем использовать алгоритмы, основанные на сходстве. Попробуем выделять группы, используя разные пороги сходства.

Как можно видеть, если использовать низкий порог сходства, подматрицы сливаются в одну, иначе они будут разделены. Оптимальный порог можно выбрать по функционалам качества кластеризации, что и определит количество, форму и структуру сгущений.

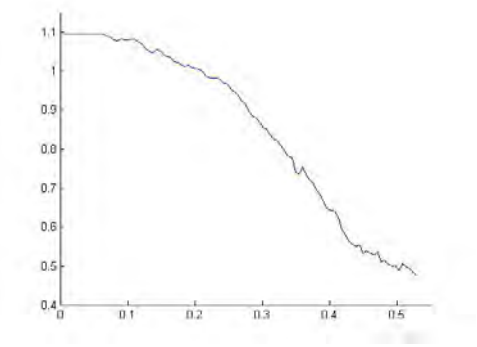

Рис. 31: Усреднение по большему числу экспериментов

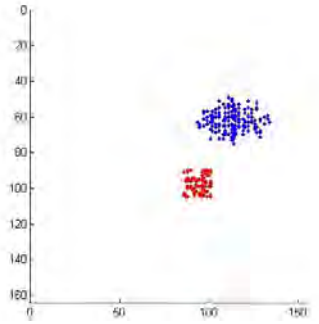

Рис. 32: Матрица A (сгущения имеют пересечения по стобцам)

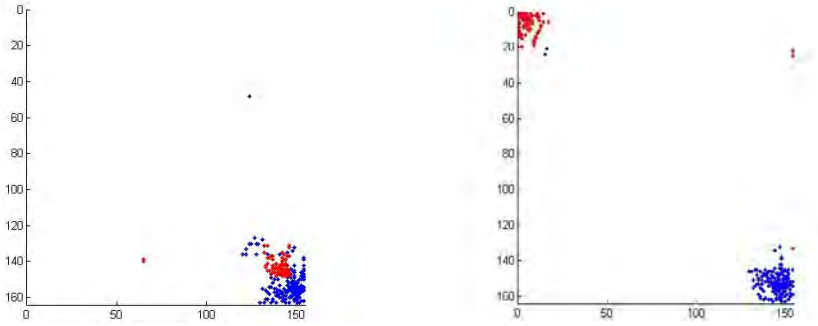

Рис. 33: Результаты выделения сгущений при низком и высоком порогах сходства

7 О задаче в рамках алгебраического подхода

7.1 Постановка задачи в рамках алгебраического подхода

Рассмотрим задачу синтеза алгоритмов A, реализующих отображения из пространства возможных начальных информаций $\mathfrak{J}_i$ в пространство возможных финальных информаций $\mathfrak{J}_f$. Множество всех отображений из $\mathfrak{J}_i$ в $\mathfrak{J}_f$ обозначим $\mathfrak{M}_*$.

$\mathfrak{M}_* = \{A|A : \mathfrak{J}_i \to \mathfrak{J}_f\}$

Применительно к задаче выделения плотных подматриц пространства начальных и финальных информаций выглядят следующим образом:

$\mathfrak{J}_i = \mathfrak{C}_{mn}(\{0,1\})$ - пространство всех матриц с элементами из множества $\{0,1\}$,

$\mathfrak{J}_f = (u,v), u \in \mathfrak{B}^m, v \in \mathfrak{B}^n$, то есть пространство финальных информаций представляет собой пару двоичных векторов размеров m и n, где i-ый элемент вектора u соответствует индикатору принадлежности i-ой строки плотной подматриц, j-ый элемент вектора v соответствует индикатору принадлежности j-го столбца плотной подматрице.

Задачи определяются структурными информациями I_s, выделяющими из $\mathfrak{M}_*$ подмножества допустимых отображений, обозначаемых $\mathfrak{M}[I_s]$.

Расширение исходного семейства алгоритмов можно осуществить с помощью корректирующих операций. Корректирующей операцией можно считать произвольную операцию над множеством $\mathfrak{M}_*$ всех отображений из пространства возможных начальных информаций $\mathfrak{J}_i$ в пространство возможных финальных информаций $\mathfrak{J}_f$. Если $\mathfrak{F}$ - некоторое семейство таких операций, то модель $\mathfrak{M}$ заменяется на «расширенную» модель $\mathfrak{F}(\mathfrak{M})$:

$\mathfrak{F}(\mathfrak{M}) = \{F(A_1,\ldots,A_p)|F \in \mathfrak{F}, (A_1,\ldots,A_p) \in \mathfrak{F}^p\}$

Применение корректирующих операций позволяет реализовать идею о совместном использовании нескольких эвристических алгоритмов при решении единственной задачи с целью устранения недостатков одних алгоритмов за счёт остальных.

Множество $\mathfrak{F}(\mathfrak{M})$ называют $\mathfrak{F}$-расширения моделей $\mathfrak{M}$, поскольку в $\mathfrak{F}$ чаще всего содержится тождественный унарный оператор, чем гарантируется выполнения включения $\mathfrak{M} \in \mathfrak{F}(\mathfrak{M})$.

Целью применения корректирующих операций является обеспечение разрешимости достаточно широкого круга задач, то есть выполнение условия $\mathfrak{F}(\mathfrak{M}) \bigcap \mathfrak{M}[I_s] \neq \varnothing$.

Рассмотрим примеры корректирующих операций для задачи выделения плотных подматриц. Пусть $U \in \mathfrak{C}_{mn}(\{0,1\})$, $A_1, A_2 \in \mathfrak{M}_*$, $A_1(u) = (u^1, v^1), A_2(u) = (u^2, v^2)$. Тогда примерами корректирующих операций могут служить:

$$F(A_1(U), A_2(U)) = F((u^1, v^1), (u^2, v^2)) = (max(u^1, u^2), max(v^1, v^2))$$

$$F(A_1(U), A_2(U)) = F((u^1, v^1), (u^2, v^2)) = (min(u^1, u^2), min(v^1, v^2))$$

Также для получения более богатого семейства алгоритмов вводится «промежуточное» по отношению к $\mathfrak{J}_i$ и $\mathfrak{J}_f$ пространство возможных оценок $\mathfrak{J}_e$. При этом корректные алгоритмы синтезируются на базе эвристических информационных моделей, т. е. параметрических семейств отображений из $\mathfrak{J}_i$ в $\mathfrak{J}_f$, представляющих собой суперпозиции отображений из $\mathfrak{J}_i$ в $\mathfrak{J}_e$, называемых алгоритмическими операторами, и отображениеями из $\mathfrak{J}_e^p$ в $\mathfrak{J}_f$.

Таким образом, $\mathfrak{M}$ определяются моделями алгоритмических операторов $\mathfrak{M}^0$, где $\mathfrak{M}^0 \in \{B|B : \mathfrak{J}_i \to \mathfrak{J}_e\}$ и решающих правил $\mathfrak{M}^1$, где $\mathfrak{M}^1 \in \{C|C : \mathfrak{J}_e \to \mathfrak{J}_f\}$ следующим образом:

$\mathfrak{M} = \mathfrak{M}^1 \cdot \mathfrak{M}^0$

Для синтеза корректных алгоритмво используются корректирующие операции F, определённые над множеством $\mathfrak{M}_*^0$. Применение данного семейства таких операций эквивалентно тому, что поиск ведётся в рамках $\mathfrak{F}$-расширения модели $\mathfrak{M}$:

$\mathfrak{F}[\mathfrak{M}] = \mathfrak{M}^1 \cdot \mathfrak{F}[\mathfrak{M}^0] = \{C \cdot F(B_1, \ldots, B_p) | C \in \mathfrak{M}^1, F \in \mathfrak{F}, B_1, \ldots, B_p \in \mathfrak{M}^0\}$

Рассмотрим возможные примеры пространства оценок, корректирующих операций, решающих правил для задачи выделения плотных подматриц.

$\mathfrak{J}_e = (x, y), x = (x_1, \ldots, x_m), x_i \in [0, 1]$ — оценка принадлежности i-ой строки плотной подматрице, $y = (y_1, \ldots, y_m), y_j \in [0, 1]$ — оценка принадлежности j-ого столбца плотной подматрице.

$B_l : \mathfrak{J}_i \to \mathfrak{J}_f, l = 1, \ldots, p$

$B_l(U) = (x^l, y^l)$

$F(B_1(U), \ldots, B_p(U)) = F((x_1, y_1), \ldots, (x_l, y_l))$;

Примерами корректирующих операций могут служить: максимум, минимум, усреднение, взвешенная сумма. В качестве решающих правил можно взять пороговые решающие правила с различными порогами для строк и столбцов.

7.2 Разрешимые задачи и корректные алгоритмы

Задача Z выделения плотной подматрицы описывается исходной матрицей $U \in \mathfrak{C}_{mn}(\{0, 1\})$ и векторами индикаторов $u \in \mathfrak{B}^m, v \in \mathfrak{B}^n$ принадлежностей строк и столбцов данной матрицы плотной подматрице.

Для задачи Z рассмотрим множества $u_0 = \{i | u(i) = 0\}$, $v_0 = \{j | v(j) = 0\}$, $u_1 = \{i | u(i) = 1\}$, $v_1 = \{j | v(j) = 1\}$. Обозначим : s_{u_0}, s_{u_1}, s_{v_0}, s_{v_1} — подстановки на множествах u_0, u_1, v_0, v_1 соответственно. σ_{u_0}, σ_{u_1}, σ_{v_0}, σ_{v_1} — множества всех подстановок на множествах u_0, u_1, v_0, v_1 соответственно. Таким образом, мы определили подстановки на множествах строк и столбцов, образующих сгущение, а также на множествах строк и столбцов, лежащих вне сгущения.

Будем говорить, что произвольный алгоритм $A : \mathfrak{J}_i \to \mathfrak{J}_f$ коммутирует с σ_{u_0}, если для $\forall s_{u_0} \in \sigma_{u_0}$ $A(s_{u_0}(U)) = (u, v)$. Коммутация алгоритма A с σ_{u_1}, σ_{v_0}, σ_{v_0} определяется аналогично.

Рассмотрим $S_1, \ldots, S_q$ — множество прецедентов (некоторых матриц, для который известны векторы $(u^1, v^1), \ldots, (u^q, v^q)$).

Пусть $\mathfrak{M}$ - множество всех отображений $A : \mathfrak{J}_i \to \mathfrak{J}_f$, I_s — система требований коммутации, которая выделяет из множества $\mathfrak{M}$ некоторое семейство отображений.

Обозначим: $\mathfrak{M}_* = \mathfrak{M}[I_s] = \{A \,|\, A(s(S_p)) = (u^p, v^p), \forall p \in 1, \ldots, q, \forall s \in \sigma_{(u^p)_0}, \forall s \in \sigma_{(u^p)_1} \forall s \in \sigma_{(v^p)_0}, \forall s \in \sigma_{(v^p)_1}\}$, то есть $\mathfrak{M}_*$ представляет собой множество алгоритмов, удовлетворяющих требованиям безошибочности и коммутации на прецедентах.

Определение. Любой алгоритм $A \in \mathfrak{M}_*$ будем называть **корректным**.

Определение. Будем говорить, что задача Z, определяемая матрицей начальной информации U, (ε, δ)-разрешима, если $\exists$ $A \in \mathfrak{M}_*$ такой, что :

$A(U) = (u', v') : \rho(U(u', v')) \geq \varepsilon$, $||u|| * ||v|| \geq \delta * m * n$.

Таким образом, задача считается разрешимой, если существует корректный алгоритм, который выделяет подматрицу плотности больше ε размеров не меньше $\delta * m * n$.

В рамках алгебраического подхода, однако,помимо понятия разрешимости используется понятие регулярности. Использование регулярности обусловлено прежде всего тем, что задачи при теоретическом исследовании рассматриваются не индивидуально, а как представители некоторых классов в известном смысле однородных задач. Отсюда вытекает, что, изучая метод решения (модель алгоритмов и т.п.), следует ставить вопрос об одновременной разрешимости задач из некоторых семейств с тем,чтобы для конкретной задачи уже из самого факта ее принадлежности такому семейству можно было делать вывод о ее разрешимости данным методом.

Определение. Будем говорить, что (ε, δ)-разрешимая задача Z, определяемая матрицей начальной информации U и парой (u', v') такой, что, $\rho(U(u', v')) \geq \varepsilon$, $||u|| * ||v|| \geq \delta * m * n$ k-регулярна, если любая задача, определяемая новой матрицей начальной информации, полу-

ченной удалением любых k единиц из подматрицы матрицы U, образованной векторами (u', v'), является (ε, δ)-разрешимой.

В рамках дальнейшей работы возможно более подробное изучение классов разрешимых и регулярных задач, а также семейств их решений.

8 Тестирование на реальных данных

8.1 Описание входных данных

В качестве используемых данных были выбраны Book-Crossing Dataset из [?], которые представляют собой оценки пользователей книг (от 1 до 10) или 0, если пользователь не чи-тал данную книгу или не пожелал оценить её. Данные представляют собой оценки 95,513 пользователей 322,199 объектов. Всего было проведено 1,048,585 оценок. Заполненность 0.0015

По данным построим матрицу «клиенты-сервисы», где в (i, j) позиции матрицы находится оценка i-ым пользователем j-ой книги. Из-за накладных расходов хранения матрицы по всем оценкам в памяти было оставлено первые 100000 оценок.

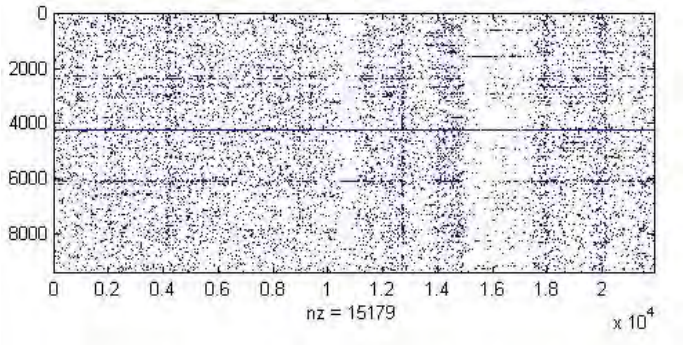

Рис. 34: Матрица, построенная по 100000 оценкам

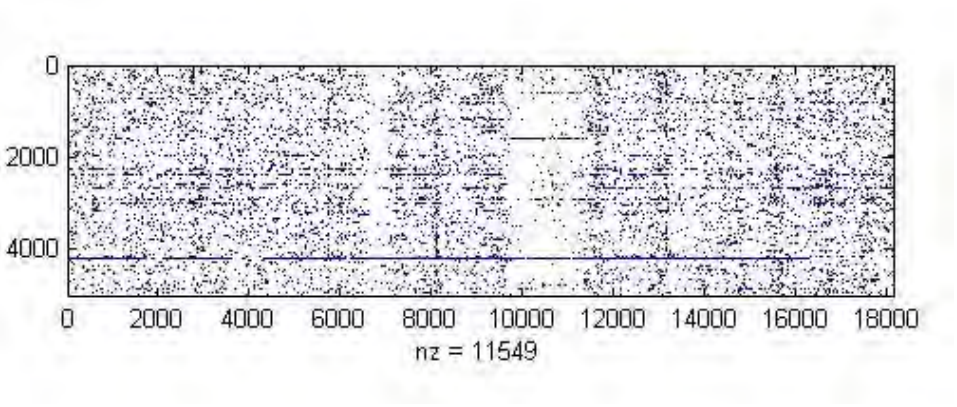

Рис. 35: Матрица, построенная по 50000 оценкам

8.2 Результаты применения алгоритмов

Будем применять алгоритмы поиска сгущений, основанные на сходстве. Изучим, как меняется время работы алгоритма в зависимости от числа оценок (следовательно, от размера матрицы). Результаты работы алгоритма, основанного на сходстве, приведены в Таблице 1.

Как можно заметить, с ростом размеров матрицы время нелинейно увеличивается. Графики зависимости времени от логарифма числа оценок и от размеров приведены на рисунке 36, 37.

Запустим градиентный алгоритм выделений сгущений. Результаты работы алгоритма приведены в Таблице 2.

Таблица 1: Время работы алгоритма выделения плотной подматрицы

Число оценок и размер матрицы	Время, с
1000 (164× 493)	0.7691701
2000 (268× 969)	1.927374
5000 (679× 2342)	2.83881
10000 (941× 4669)	5.163881
20000 (2180× 8586)	18.05481
30000 (3445× 11997)	47.534711
50000 (5064× 18125)	159.499643
75000 (6693× 51182)	460.263267
100000 (9424× 31867)	843.839536890

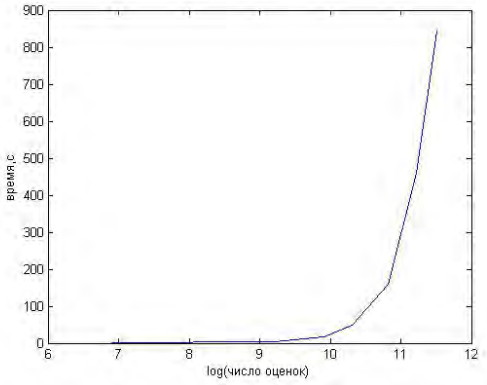

Рис. 36: График зависимости времени работы от логарифма числа оценок

Как можно видеть, время работы градиентного алгоритма существенно меньше.

Градиентный алгоритм:

Вход: Матрица A

Выход: набор плотных подматриц

1: выбрать в качестве начального приближения некоторое сгущение
2: **повторять**
3: выбрать и присоединить к текущей матрицы строку или столбец, не уменьшающие плотность
4: **пока** множество строк или столбцов, не уменьшающих плотность, непусто

Протестируем алгоритм с разными порогами сходства на матрице размера 9424×31867, число ненулевых элементов 11549. Результаты приведены в таблице 3.

Как можно видеть, с ростом порога сходства плотность подматрицы максимального размера увеличивается, однако уменьшается её размер (рис. 38,39).

Предположим, у нас имеется некоторое плотное подмножество строк и столбцов плотности ρ. Будем присоединять строки и столбцы к данному сгущению градиентным алгоритмом (на каждом шаге к сгущению присоединяется строка или столбец, дающие наилучшее изменение среди оставшихся). Графики, иллюстрирующие данные эксперименты, приведены на рисунках 42, 43.

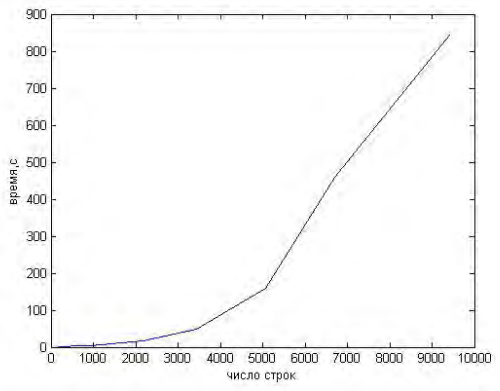

Рис. 37: График зависимости времени работы от размеров

Таблица 2: Время работы градиентного алгоритма выделения плотной подматрицы

Число оценок и размер матрицы	Время, с
1000 (164× 493)	0.33429
2000 (268× 969)	0.94328
5000 (679× 2342)	1.23490
10000 (941× 4669)	2.5702
20000 (2180× 8586)	7.43998
30000 (3445× 11997)	16.2390
50000 (5064× 18125)	29.32983
75000 (6693× 51182)	57.349913
100000 (9424× 31867)	94.43905

Таблица 3: Результаты работы алгоритма выделения плотных подматриц

Пороги сходства по строка и столбцам	Число строк	Число столбцов	Плотность
(2,2)	313	481	0.0158
(2,3)	132	90	0.0573
(2,4)	36	24	0.2174
(2,5)	18	6	0.4259
(2,6)	13	3	0.7179
(3,2)	178	481	0.0242
(3,3)	62	90	0.0935
(3,4)	24	24	0.2413
(3,5)	6	6	0.5833
(3,6)	13	3	0.7179
(4,2)	115	481	0.0334
(4,3)	37	90	0.1303
(4,4)	12	24	0.3333

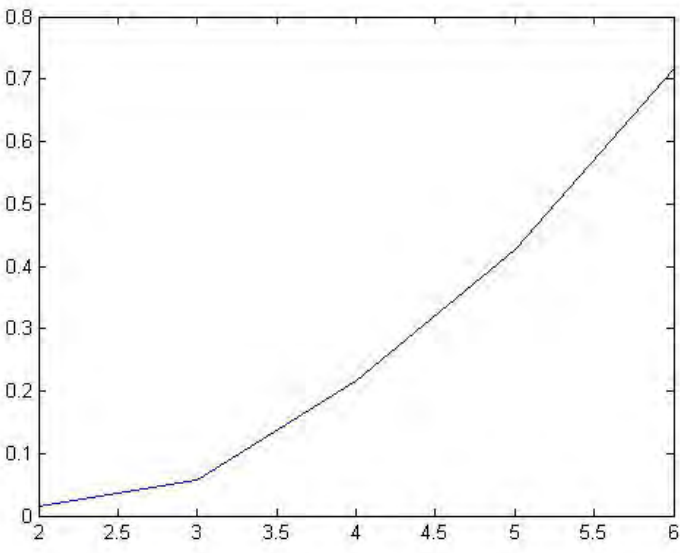

Рис. 38: Зависимости плотности сгущения с увеличением порога

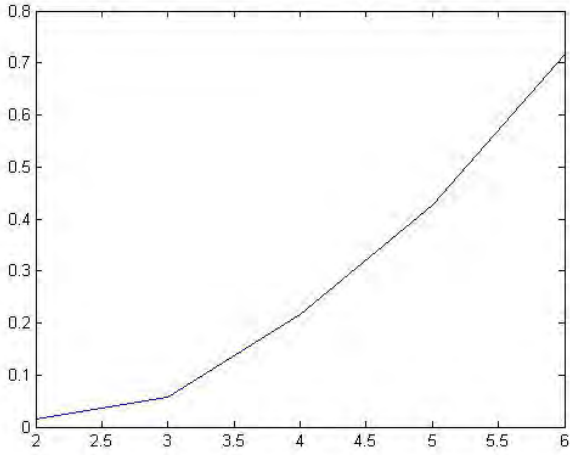

Рис. 39: Зависимости размеров сгущения с увеличением порога

Рис. 40: Пример выделенных плотных подматриц

Рис. 41: Пример выделенных плотных подматриц

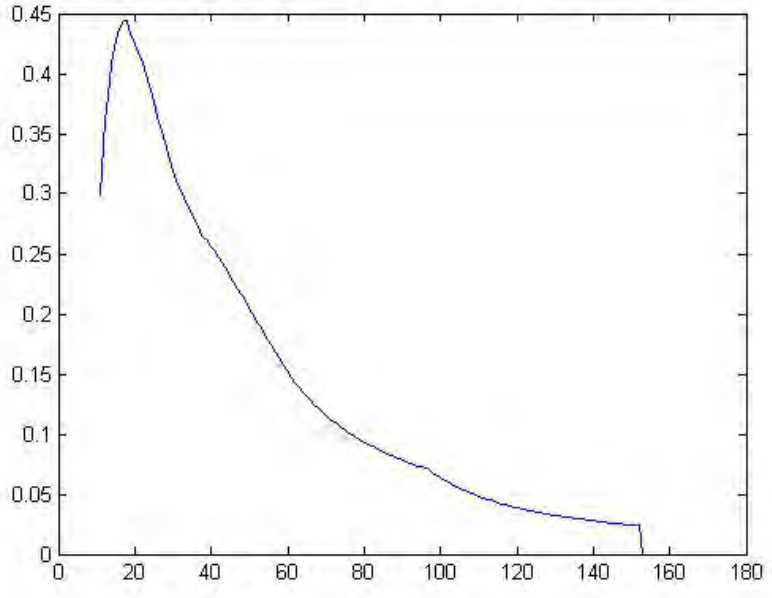

Рис. 42: Конфигурация и график зависимости плотности от числа строк и столбцов в сгущении

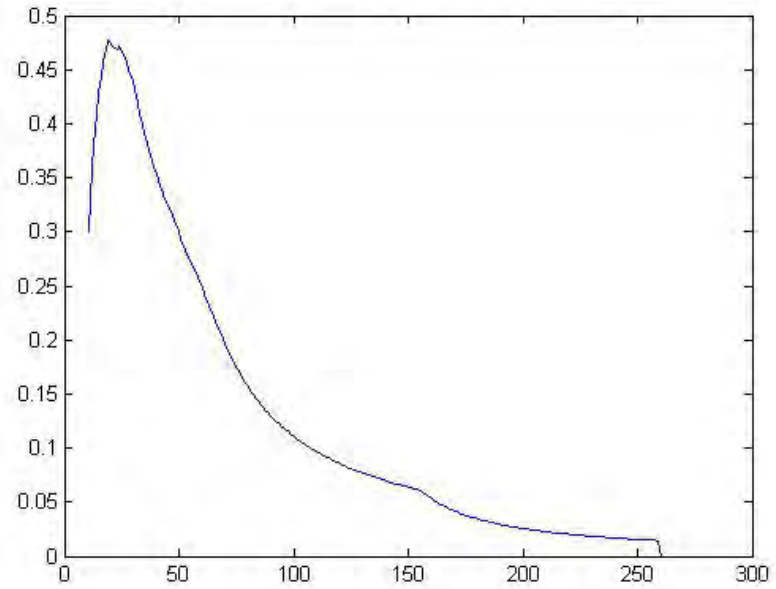

Рис. 43: Конфигурация и график зависимости плотности от числа строк и столбцов в сгущении

9 Заключение

В рамках данной работы были исследованы алгоритмы выделения плотных подматриц, а также методы оценки их качетва, сложность задачи. Задача рассмотрена с точки зрения алгебраического подхода, исследована возможная прецедентная информация, рассмотрены некоторые семейства алгоритмов, классы разрешимых и регулярных задач. Алгоритмы протестированы на модельных и реальных данных.

Список литературы

[1] К.В. Воронцов *«Математические методы обучения по прецедентам»*.

[2] Дьяконов, А. Г *«Анализ данных, обучение по прецедентам, логические игры, системы WEKA, RapidMiner и MatLab (практикум на ЭВМ кафедры математических методов прогнозирования)»* МАКСПресс, 2010.

[3] Журавлёв Ю.И *«Избранные научные труды»* - М.: «Магистр», 1998.– 420с.

[4] Журавлёв Ю.И *«Корректные алгебры над множествами некорректных (эвристических) алгоритмов.»* - Кибернетика. — 1977 - 1978

[5] Рудаков К.В. *«Об алгебраической теории универсальных и локальных ограничений для задач классификации»* - Распознавание, классификация, прогноз. М.: Наука, 1989

[6] Рудаков К.В. *«Полнота и универсальные ограничения в проблеме коррекции эвристических алгоритмов классификации* - Кибернетика. 1987. N»

[7] Роджерс Д *«Алгоритмические основы машинной графики»*М.: Мир, 1989

[8] Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. *«Алгоритмы. построение и анализ»* - 2005

[9] Yizong Cheng and George M. Church *«Biclustering of Expression Data»* - 2000

[10] Источник данных Book-Crossing Dataset *«http://www2.informatik.uni-freiburg.de/cziegler/BX/»*

[11] Задача Netflix *«http://www.netflixprize.com»*