УДК 519.254

БЫСТРЫЙ АЛГОРИТМ КЛАСТЕРНОГО АНАЛИЗА k-MEDOIDS

И. Н. Дмитриев

Федеральное учебно-методическое объединение в сфере высшего образования по УГСН 10.00.00 Информационная безопасность, г. Москва, Россия

Рассмотрена новая реализация FKM известного алгоритма k-medoids, основанная на общеизвестной PAM-реализации и использующая новую эвристику выбора центров кластеров, методику CLARA, а также предварительное прореживание L-SPAR, что позволило перейти от квадратичной вычислительной сложности реализации к линейной и снизить временные затраты на обработку реальных данных в среднем в 16 раз.

Ключевые слова: *быстрый алгоритм кластерного анализа, PAM-реализация k-medoids, методика CLARA, прореживание L-SPAR.*

DOI 10.17223/20710410/39/11

FAST ALGORITHM OF CLUSTER ANALYSIS k-MEDOIDS

I. N. Dmitriev

Federal Educational and Methodological Association in Information Security, Moscow, Russia

E-mail: i.n.dmitriev@mail.ru

The algorithm k-medoids (KM) is widely used for clustering graphs. The following modifications of it by adding some procedures have been researched in a computer experiments: CKM = KM + CLARA (Cluster LARge Application), LSKM = KM + + L-SPAR (Local SPARsification), BKM = KM + CLARA + L-SPAR, NKM = = KM + NH (New Heuristic choosing cluster center), CNKM = NKM + CLARA, LSNKM = NKM + L-SPAR, FKM = NKM + CLARA + L-SPAR. The experiment results show that L-SPAR procedure allows to decrease the average running time for LSKM by 5.3 %, BKM by 6.1 %, LSNKM by 8.1 %, FKM by 8.3 % and to improve the clustering quality by 2 %, 2.1 %, 3 %, 3.3 % respectively. Besides, the number of iterations in NKM is twice more than in KM, but the running time of NKM is an order less than of KM, the running time of CKM is two thirds of the running time of KM, the computer complexity of CNKM linearly depends on the graph size, and the application of CLARA does not diminish it appreciably. NH allows 16 times decreasing the computer complexity of KM without loosing the clustering quality. The experiments were conducted on data arrays that represented graphs of social networks YouTube, Live Journal, and the shop Amazon.

Keywords: *fast algorithm of cluster analysis, PAM, CLARA, Local Graph Sparsification.*

Введение

Известный алгоритм k-medoids является модификацией классического алгоритма кластеризации k-means и предназначен для решения задач выделения групп объектов (кластеров) в случаях, когда проводится кластеризация объектов без использования свойств линейного пространства. Такие задачи возникают, когда используется

специфическая мера близости объектов (не расстояние), например при кластеризации неориентированных графов. В этом случае, в отличие от k-means, центром кластера может быть не любая точка признакового пространства (центроид), а только точка, принадлежащая кластеризуемой выборке — медоид.

Неориентированные графы используются для представления информации о фактах взаимодействия или взаимосвязи объектов. Из них легко получить количество взаимодействий с объектом, как с каждым в отдельности, так и совместных взаимодействий для каждой пары объектов. В частности, задачу выделения новых групп в данных сетевого трафика можно рассматривать как задачу выделения кластеров в графе взаимодействия узлов сети и дальнейшего анализа этих кластеров с использованием контентных характеристик.

На концептуальном уровне кластером неориентированного графа называется такая группа вершин, в которой внутригрупповые связи гораздо плотнее межгрупповых [1]. В настоящей работе используется предположение о структуре кластеров в графе, согласно которому каждая вершина графа входит в один и только один кластер. Отметим, что из предположения следует, что разбиение полностью покрывает граф. Тогда можно говорить о поиске кластеров в графе как о задаче поиска разбиения множества вершин на подмножества, которое минимизирует некоторый функционал.

Для решения задачи выделения кластеров в графах предназначена PAM-реализация (Partitioning Around Medoids) алгоритма k-medoids [2]. В работе рассмотрен быстрый алгоритм k-medoids (Fast K-Medoids, FKM) — модификация PAM, в которой предложена новая эвристика выбора следующего медоида, позволяющая снизить вычислительную сложность алгоритма, дополненная механизмом предварительного прореживания L-SPAR и методикой кластеризации больших массивов данных CLARA (CLustering LARge Applications). В результате экспериментальной оценки временных характеристик и показателей качества FKM относительно PAM-реализации алгоритма k-medoids, также дополненного CLARA и L-SPAR (далее — Basic K-Medoids, BKM), на реальных графах, построенных на базе сетевых ресурсов YouTube, Live Journal и Amazon, показано, что FKM не только не уступает в качестве разбиения BKM, но и позволяет сократить временные затраты на обработку массивов данных.

1. Быстрый алгоритм кластерного анализа k-medoids

Для описания PAM введем понятие модулярности, которая используется в качестве функции потерь при оценке «удачности» выбора нового медоида, а также при проверке критерия останова алгоритма.

Пусть $G = (V, E)$ — неориентированный граф с матрицей смежности $A = (a_{ij})$, где V — множество вершин, E — множество рёбер, и $C = \{C_1, C_2, \dots C_k\}$ — разбиение вершин графа на кластеры.

Определение 1. Модулярностью Гирвана — Ньюмана [3] для графа $G = (V, E)$ называется функционал вида

$$Q(C) = \frac{1}{2\,|E|}\sum_{i=1}^{|V|}\sum_{j=1}^{|V|}\left(a_{ij} - \gamma\frac{\deg(v_i)\deg(v_j)}{2\,|E|}\right)\sigma(h_C(v_i), h_C(v_j)),$$

где $\deg(v_i)$ — степень вершины графа v_i; $h_C(v_i)$ — номер кластера, в который попала вершина v_i при разбиении C; σ — символ Кронекера; $\gamma > 0$ — константа, позволяющая управлять размером кластеров. Выбор $\gamma < 1$ приводит к увеличению размера кластеров в разбиении, при $\gamma > 1$ размеры кластеров уменьшаются, а их количество возрастает.

Функционал $Q(C)$ не является непрерывным и задача его оптимизации дискретная. Для поиска глобального оптимума используют приближённые алгоритмы. Некоторые из них действительно оптимизируют значение функционала, другие по значению модулярности выбирают наилучшее решение из найденных, то есть не дают гарантий локальной оптимальности решения.

Для определения степени близости двух вершин в методике CLARA, способе предварительного прореживания L-SPAR, а также на этапе распределения вершин по кластерам в реализации PAM используется коэффициент Жаккарда. Применительно к неориентированному графу G для двух вершин v_i и v_j этот коэффициент имеет следующий вид: $\mathrm{sim}(v_i, v_j) = \dfrac{|\mathrm{Adj}\,(v_i) \cap \mathrm{Adj}\,(v_j)|}{|\mathrm{Adj}\,(v_i) \cup \mathrm{Adj}\,(v_j)|}$, где $\mathrm{Adj}(i)$ — множество соседей вершины i.

Будем понимать под медоидом объект, принадлежащий набору данных или кластеру, отличие которого от других объектов в нём минимально. Медоиды близки по смыслу центроидам, но, в отличие от них, являются объектами, принадлежащими кластеру, и, как правило, используются в тех случаях, когда невозможно вычислить средние координаты или центр масс кластера.

PAM является одной из реализаций k-medoids (алгоритм 1) и представляет собой «жадный» алгоритм с простейшей эвристикой (проход по всем вершинам, шаг 4 алгоритма).

Алгоритм 1. PAM-реализация алгоритма k-medoids

Вход: Граф G, заданное число кластеров k.
Выход: Оптимальное разбиение на k кластеров (оптимум модулярности).
Медоид для каждого кластера

1: ***Инициализация***
Случайным образом выбираем k вершин для множества медоидов M.
Для каждой вершины находим ближайший (по Жаккарду) медоид, формируя разбиение на кластеры $C = \{C_1, C_2, \ldots C_k\}$.
$minCost \leftarrow$ значение модулярности для исходного разбиения.
2: **Повторять**
3: **Для всех** $m \in M$
4: **Для всех** $v \in C_m$ и $v \neq m$
5: перемещаем медоид в v;
6: строим очередное разбиение с новым медоидом;
7: $cost \leftarrow$ модулярность для очередного разбиения.
8: **Если** $cost < minCost$, **то**
9: $m' \leftarrow m$, $minCost \leftarrow cost$.
10: $m \leftarrow m'$
11: **Пока** не перестанет изменяться $minCost$.

Оценим вычислительную сложность PAM-реализации алгоритма k-medoids. На каждой итерации перебирается $n - k$ вершин графа (здесь n — общее количество вершин), в которые перемещается соответствующий медоид. Для каждой такой замены нужно пересчитать расстояния от всех точек до медоидов. Если все попарные расстояния между точками помещаются в память, то этот этап займёт $(n - k)\,k$ действий. Оптимальная замена требует ещё $(n - k)\,k$ действий. Таким образом, сложность одной

итерации в худшем случае равна $(n-k+1)(n-k)k = \mathrm{O}(n^2k)$, то есть вычислительная сложность реализации $\mathrm{O}(n^2kI)$, где I — количество итераций.

Одним из слабых мест PAM является неэффективный поиск новых кандидатов на места медоидов. Перебор всех имеющихся вершин в графе нельзя назвать оптимальным, попытаемся сократить число кандидатов для перебора. Введём понятие множества замен кластера C_m с медоидом в m.

Определение 2. Множество $Z_m^s \subset C_m \setminus \{m\}$, состоящее из s вершин, будем называть множеством замен кластера C_m.

Для построения множества Z_m^s могут быть использованы следующие техники:

1) выбор s случайных вершин из C_m с равной вероятностью;
2) выбор s случайных вершин из C_m с вероятностью, пропорциональной степени вершины;
3) выбор $s/2$ пар вершин из C_m с наибольшим значение коэффициента Жаккарда;
4) выбор s вершин из «ближайших» соседей медоида m.

Для того чтобы однозначно сказать, какая из техник позволит сократить число итераций, повысить качество итогового разбиения или прийти к глобальному минимуму функционала модулярности, необходимо провести дополнительные эксперименты. В данной работе для построения множества Z_m^s использована самая простая техника: равновероятный выбор s случайных вершин из кластера.

Суть новой эвристики заключается в сокращении множества для выбора нового медоида со всего графа до множества замен кластера C_m. Это позволяет искать оптимальную замену не по всем вершинам графа, а по одному кластеру, перебирая не все вершины внутри этого кластера, а лишь s из них. В этом случае сложность одной итерации равна $(n-k)k(s+1) = \mathrm{O}(nks)$ — она линейна по числу вершин, а сложность всей реализации составляет $\mathrm{O}(nksI)$, где $s \ll n/k$.

Исследуем, как повлияет новая эвристика на качество результатов, получаемых при работе алгоритма, в зависимости от количества элементов s в множестве замен. В [4] приводится пример алгоритма, в котором имеется параметр с похожим на s смыслом. В работе доказывается, что снижение значения параметра s до 2 и даже до 1 не ухудшает различные метрики качества кластеров.

Для увеличения максимального размера графа, который может обработать реализация алгоритма k-medoids, применим методику обработки данных большого объёма, называемую CLARA [2]. Применительно к задаче кластеризации графа методика CLARA имеет следующий вид: из исходного графа выбирается подмножество вершин и кластеризуется подграф, образованный этими вершинами, затем, в предположении связности графа, оставшиеся вершины распределяются по ближайшим медоидам из подграфа.

Вся сложность модификации CLARA состоит в последовательном прогоне на разных подмножествах вершин и выборе оптимального из полученных результатов. За счёт этого предполагается компенсировать ущерб от исключения части информации при каждом отдельном прогоне, а также избежать «застревания» в локальном минимуме (в качестве меры оптимальности при выборе результата используется модулярность).

Существуют следующие техники получения подграфа:

1) выбор случайных вершин с равной вероятностью;
2) выбор случайных вершин с вероятностью, пропорциональной степени вершины в исходном графе;

3) выбор вершин с наибольшей степенью;
4) выбор пар вершин с наибольшим коэффициентом Жаккарда.

Для того чтобы установить степень влияния техники на конечный результат методики CLARA, необходимо провести дополнительный эксперимент. В рамках данной работы эксперимент не проводился, поэтому предпочтение отдано первой технике, как наиболее простой и очевидной.

Выбор объёма подграфа для CLARA представляет собой компромисс между качеством и скоростью. Чем больше кластеров присутствует в данных и чем меньше они по размеру, тем больше вероятность того, что некоторые кластеры могут не попасть в выборку. Если разница в размерах кластеров большая, то такой подход вообще противопоказан. Если же кластеры в данных сравнимы по размеру, то будем брать подграф, состоящий из не менее чем $n/2$ вершин. Данное решение принято на основании результатов экспериментов, проведенных над модификацией CLARA для PAM в [2].

CLARA изначально предназначалась для уменьшения времени выполнения алгоритмов кластеризации, чья вычислительная сложность в зависимости от числа вершин растёт быстрее, чем линейно, т. е. когда выполнить алгоритм несколько раз на подграфе быстрее, чем один раз на полном графе. Поэтому выигрыш модификации CLARA по времени заметен на PAM-реализации алгоритма k-medoids, а при использовании реализации с улучшенной эвристикой (имеющей линейную сложность) выигрыш минимальный.

Следующий шаг по улучшению PAM-реализации с новой эвристикой и методикой CLARA называется L-SPAR (Local SPARsification), он подробно описан в [5]. Этот метод предобработки графа перед кластеризацией «прореживает» граф путём удаления части рёбер (но не вершин), не разрушая, как правило, его связность.

Рассмотрим подробнее алгоритм L-SPAR. Сначала для каждой вершины i все рёбра (i, j) сортируются в порядке убывания $\mathrm{sim}(v_i, v_j)$, после чего хвост списка включается в множество для фильтрации. Каждая следующая вершина даёт своё множество рёбер для фильтрации, которое объединяется с уже существующим. Когда все вершины обработаны, все рёбра полученного множества удаляются из графа. Авторы метода предлагают включать в список «выживших» $\min\{1, d_i^e\}$ рёбер, где d_i — степень вершины i и $0 \leqslant e \leqslant 1$. Если ребро попало в список «выживших» хотя бы для одной из своих вершин, оно «выживает». Таким образом, не образуется новых изолированных вершин и, если исходный граф был связным, разреженный также останется связным.

Доказано, что при применении L-SPAR сохраняется степенной закон распределения степеней вершин в графе, приводящий к феномену «тесного мира» («small world»), когда кратчайший путь между двумя вершинами в среднем имеет очень маленькую длину. В то же время степень прорежения в L-SPAR зависит от плотности графа в непосредственной близости от вершины и сохраняет структуру как плотных, так и разреженных сообществ.

Данный метод предобработки графа можно использовать при любом методе кластеризации. Наибольший эффект может быть достигнут для алгоритмов, чья сложность зависит только от числа рёбер, но не вершин, так как сокращается именно число рёбер. В этом плане реализации алгоритма k-medoids не получат глобального прироста в производительности: его сложность зависит только от числа вершин, однако при более отчетливой структуре групп может уменьшиться число итераций, необходимых для сходимости.

2. Сравнение реализаций алгоритма *k*-medoids

Перейдем к результатам экспериментов, проведённых над описанными реализациями алгоритма k-medoids. Обозначения реализаций приведены в табл. 1.

Таблица 1

Обозначения рассмотренных реализаций

Код	Описание
KM	PAM-реализация алгоритма k-medoids
CKM	KM с методикой CLARA
LSKM	KM с предварительным прореживанием рёбер L-SPAR
BKM	KM с методикой CLARA и прореживанием рёбер L-SPAR
NKM	PAM-реализация алгоритма k-medoids с новой эвристикой
CNKM	NKM с методикой CLARA
LSNKM	NKM с предварительным прореживанием рёбер L-SPAR
FKM	NKM с методикой CLARA и прореживанием рёбер L-SPAR

Эксперименты проводились на массивах данных, представляющих собой размеченные графы социальных сетей YouTube, Live Journal, а также магазина Amazon. Рассмотрим подробнее наборы данных, используемых для эксперимента [6].

Amazon — граф, построенный на основе данных о товарах магазина Amazon.com, в качестве вершины выступает товар магазина, при этом две вершины связаны, если известен факт совместной покупки товаров одним покупателем.

Live Journal — граф социальной сети LiveJournal.com, в котором в качестве вершин выступают пользователи, а в качестве связи — взаимная дружба.

YouTube — граф социальной сети Youtube.com, в котором вершинами являются пользователи, связь между ними определяется наличием подписки одного на другого.

Данные массивы обрабатывались не целиком — для кластеризации выбиралось некоторое количество групп (группы определяются разметкой графа), чтобы число вершин объединения этих групп соответствовало наперёд заданному порядку, затем строился граф с вершинами и связями, входящими только в эти группы (табл. 2).

Таблица 2

Описание характеристик данных, использованных в экспериментах

Массив	Код	Порядок количества вершин 10^l	Количество кластеров k
Amazon	A3	$l = 3$	176
	A4	$l = 4$	1635
	A5	$l = 5$	15378
Life Journal	LJ3	$l = 3$	87
	LJ4	$l = 4$	765
	LJ5	$l = 5$	8104
YouTube	Y3	$l = 3$	3
	Y4	$l = 4$	22
	Y5	$l = 5$	176

Из табл. 2 видно, что в массиве Amazon преобладают кластеры небольшого объёма, а в массиве YouTube, наоборот, предпочтение отдано кластерам, имеющим большой объём. Такое различие в размере кластеров при проведении экспериментов позволит сделать вывод о том, как влияет объём кластера на количество итераций, скорость работы алгоритма, а также на качество полученной кластерной структуры.

Эксперименты проводились по трём направлениям:

1) сравнение количества итераций для KM, NKM, LSKM и LSNKM;
2) сравнение времени работы всех описанных реализаций алгоритма k-medoids;
3) сравнение качества получаемых разбиений графа на кластеры для всех описанных реализаций алгоритма k-medoids.

Для определения мер близости результатов кластеризаций рассмотрим множество вершин $V = \{v_1, \ldots, v_n\}$ из n объектов и два различных разбиения этого множества на подмножества (кластеры): $C^{(1)} = \{C_1^{(1)}, \ldots, C_k^{(1)}\}$ и $C^{(2)} = \{C_1^{(2)}, \ldots, C_l^{(2)}\}$, где $C_i^{(1)} \subset V$, $C_j^{(2)} \subset V$, $\bigcup\limits_{i=1}^{k} C_i^{(1)} = V = \bigcup\limits_{j=1}^{l} C_j^{(2)}$, $C_i^{(1)} \cap C_{i'}^{(1)} = \varnothing$ и $C_j^{(2)} \cap C_{j'}^{(2)} = \varnothing$ для $i, i' = 1, \ldots, k$ и $i \neq i'$, $j, j' = 1, \ldots, l$ и $j \neq j'$. Пересечения кластеров, входящих в указанные разбиения, можно представить с помощью таблицы сопряжённости $T_{k\times l}$ (табл. 3). Её элементы n_{ij} суть числа вершин, общих для $C_i^{(1)}$ и $C_j^{(2)}$, $i = 1, \ldots, k$ и $j = 1, \ldots, l$. Дополнительно в $T_{k\times l}$ вводятся столбец и строка сумм основных элементов таблицы по строкам и столбцам, которые соответствуют мощностям множеств $C_i^{(1)}$ и $C_j^{(2)}$ и представляются в виде $n_{i\circ} = \sum\limits_{j=1}^{l} n_{ij}$ и $n_{\circ j} = \sum\limits_{i=1}^{k} n_{ij}$ соответственно.

Таблица 3

Таблица сопряженности $T^{k\times l}$

Номера кластеров	1	2	…	l	Суммы
1	n_{11}	n_{12}	…	n_{1l}	$n_{1\circ}$
2	n_{21}	n_{22}	…	n_{2l}	$n_{2\circ}$
…	…	…	…	…	…
k	n_{k1}	n_{k2}	…	n_{kl}	$n_{k\circ}$
Суммы	$n_{\circ 1}$	$n_{\circ 2}$	…	$n_{\circ l}$	$n_{\circ\circ} \equiv n$

Для оценки конечного разбиения будем использовать F-меру. Эта мера пришла из области оценки качества информационного поиска. Будем трактовать некоторое подмножество (кластер) разбиения вершин графа $C^{(1)}$ как результат информационного запроса, а определённое подмножество (класс) разбиения $C^{(2)}$ — как совокупность вершин графа, соответствующих этому же запросу (эталон). Тогда для j-го кластера и i-го класса полнота R_{ij} и точность P_{ij} информационного поиска суть величины

$$R_{ij} = \frac{n_{ij}}{n_{i\circ}}, \quad P_{ij} = \frac{n_{ij}}{n_{\circ j}}.$$

Для j-го кластера и i-го класса F-мера характеризуется величиной

$$F_{ij} = \left(\frac{1}{2}\frac{1}{P_{ij}} + \frac{1}{2}\frac{1}{R_{ij}}\right)^{-1}.$$

Для всех кластеров F-мера определяется следующим образом:

$$F = \sum_i \tfrac{n_{i\circ}}{n} \max_j F_{ij}.$$

Из определения F_{ij} следует, что $F_{ij} \leqslant 1$ и F_{ij} принимает максимальное значение, когда есть полное соответствие между разбиениями $C^{(1)}$ и $C^{(2)}$.

Для каждого эксперимента реализация алгоритма запускалась несколько раз, затем результаты усреднялись по количеству запусков. Компьютер, на котором производились эксперименты, имеет следующие характеристики: Intel Core M, 4 Гб RAM,

256 Гб SSD, Windows 10 Professional. Алгоритмы реализованы на языке Java, версия JRM 1.8.0.111.

Оценим количество итераций для реализаций KM и NKM в случае графа с прореженными рёбрами и без прореживания. Для этого проведём эксперимент на выборках из массивов Amazon, Live Journal, YouTube с количеством вершин порядка 10^4. Результаты представлены в виде диаграммы, на которой высота столбца отражает количество итераций (рис. 1).

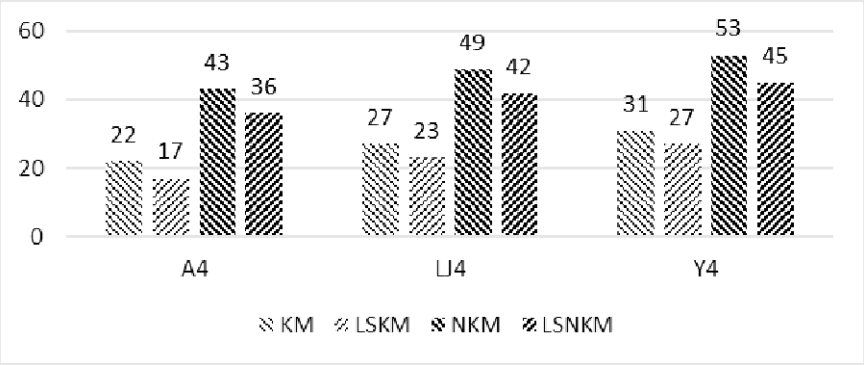

Рис. 1. Число итераций, необходимых для остановки алгоритма

Из эксперимента видно, что для реализации NKM необходимо почти в 2 раза больше итераций, чем для реализации KM. Особенно это проявляется при малом объёме кластеров в массиве Amazon. В массивах с большими кластерами эта разница менее выражена, но всё равно большая. Видно также, что применение предварительного прореживания массивов данных позволяет сократить число итераций, необходимых для успешного завершения работы. Это свидетельствует о том, что локальное прореживание позволяет избавиться от лишнего «шума» среди связей объектов и получить более ясную кластерную структуру.

Оценим временные показатели работы реализаций алгоритма на данных, не подвергавшихся никакой предварительной обработке. При проведении эксперимента время фиксировалось в секундах. Так как разброс значений времени работы имеет достаточно большой диапазон, на графике представлен десятичный логарифм от значений времени работы реализации в соответствующем эксперименте (рис. 2).

Несмотря на то, что реализация NKM требует большего числа итераций, чем базовая реализация KM, она показала время разбиения графа на кластеры на порядок меньше, чем KM. Данный результат связан с введением новой эвристики, позволяющей снизить вычислительную сложность алгоритма.

Реализация CKM позволила сократить время работы в среднем на треть, однако для реализации CNKM время работы на тех же данных сократилось менее чем на процент. Такой контраст вызван тем, что CKM имеет квадратичную сложность и многократная обработка выборок меньшего объема позволяет сократить общее время работы алгоритма, чего нельзя сказать о модификации CNKM: её сложность линейно зависит от числа элементов и применение методики CLARA не позволяет существенно снизить время обработки исходного графа.

Проведём аналогичный эксперимент с данными, которые предварительно прошли прореживание с использованием метода L-SPAR. Для наглядности результатов на графике вместо значения времени работы будем отображать долю от времени, которое

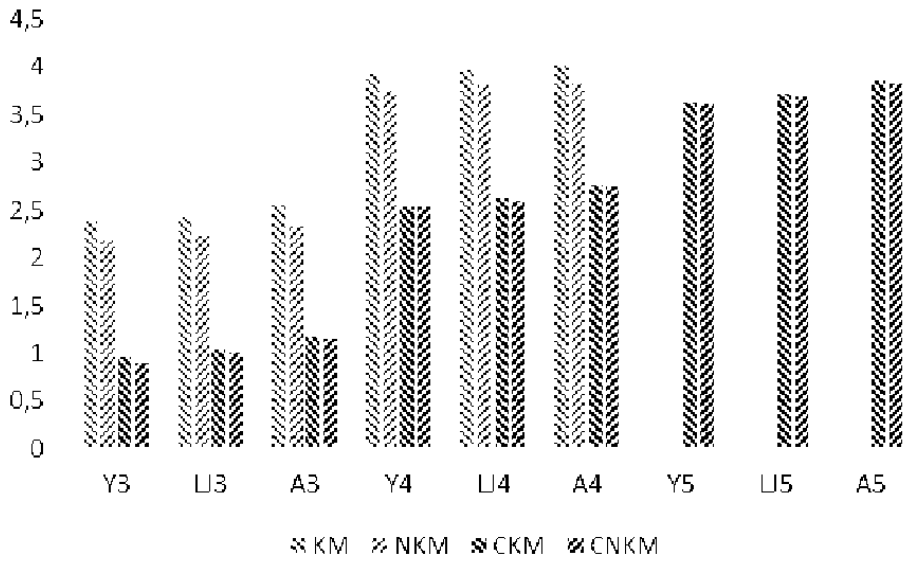

Рис. 2. Десятичный логарифм временных затрат реализаций на различных выборках

потребовалось на обработку данных без предварительного прореживания рёбер графа (рис. 3).

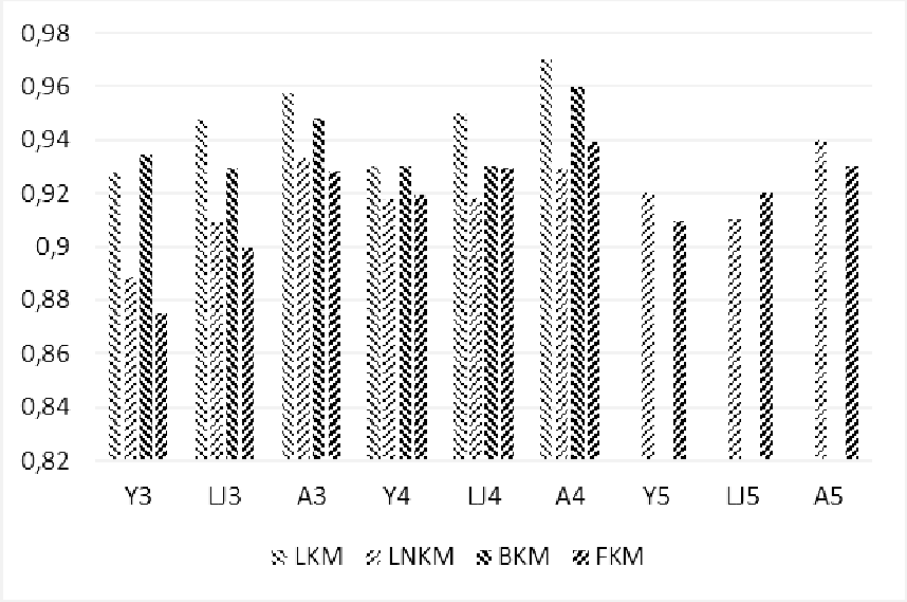

Рис. 3. Доля от времени работы реализаций с прореживанием данных и без него

Из результатов эксперимента следует, что среднее время работы сократилось для LSKM на 5,3 %, BKM на 6,1 %, LSNKM на 8,1 %, FKM на 8,3 %. Реализация LSNKM из-за прореживания получает больший процент сокращения времени обработки данных по сравнению с LSKM, так как граф без «шумных» рёбер имеет более чёткую кластерную структуру и сокращается число итераций для этой реализации. На данных, где объём кластеров больше, положительное действие прореживания рёбер графа усиливается из-за сокращения шума среди связей графа.

Последний аспект, по которому проводилось сравнение реализаций, — качество получаемого разбиения графа. Для оценки качества использована F-мера (рис. 4).

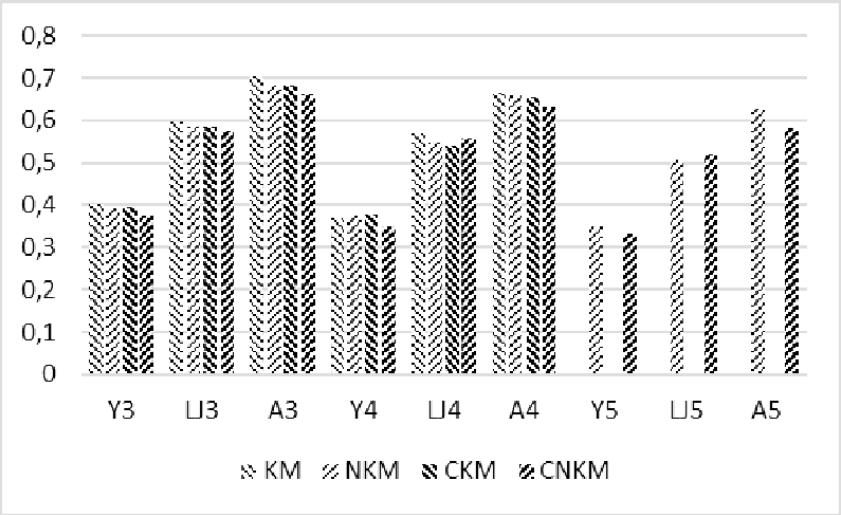

Рис. 4. Значение показателя качества при обработке массивов различными реализациями алгоритма k-medoids

Реализация с новой эвристикой NKM позволяет получить разбиение графа, по качеству не уступающее исходной реализации KM (в рамках статистической погрешности). Для реализаций с методикой CLARA характерно меньшее значение F-меры, чем для аналогичных модификаций без CLARA, в связи с тем, что алгоритмом обрабатывается только часть выборки, а остальные объекты распределяются с использованием коэффициента Жаккарда.

Рассмотрим аналогичный эксперимент с данными, предварительно обработанными методикой прореживания L-SPAR. Как и в предыдущем примере, на графике будем отображать не значение F-меры в конкретном эксперименте, а долю этого значения от показателя качества, полученного на данных без предварительного прореживания (рис. 5).

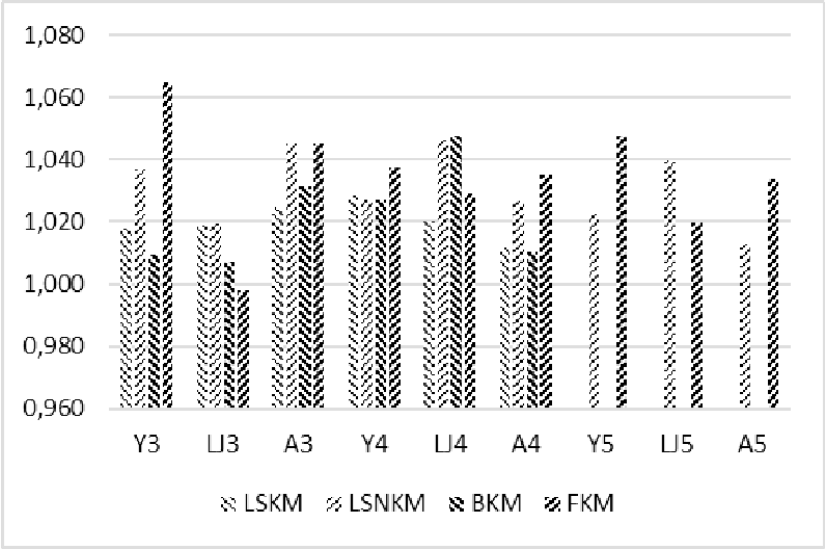

Рис. 5. Доля, на которую увеличилось качество разбиения графа после процедуры прореживания

Из результатов эксперимента следует, что среднее качество разбиения графа улучшилось для LSKM на 2 %, BKM на 2,1 %, LSNKM на 3 %, FKM на 3,3 %. Чёткая кластерная структура, полученная после прореживания «шумных» рёбер графа, позволяет реализациям повысить качество получаемых разбиений.

Заключение

Рассмотрен алгоритм кластеризации неориентированных графов на основе известного алгоритма k-medoids, а также его реализации с новой эвристикой и методикой CLARA. Кроме этого, исследован вопрос использования механизма предварительного прореживания рёбер графа, позволяющего избавиться от «шума» в множестве связей графа. В экспериментах показано, что новая эвристика позволяет существенно (в среднем в 16 раз) снизить вычислительную сложность алгоритма k-medoids и при этом сохранить качество разбиения графа.

Предложенная реализация алгоритма с использованием методики CLARA позволила на треть сократить время обработки данных для PAM-реализации алгоритма k-medoids, однако для реализации с новой эвристикой использование этой методики не дало ощутимого сокращения времени обработки графа. Сокращение времени работы привело к потере качества и в том и в другом случае.

Предварительное прореживание графа при помощи методики L-SPAR позволило сократить среднее время работы для LSKM на 5,3 %, BKM на 6,1 %, LSNKM на 8,1 %, FKM на 8,3 %. Кроме этого, использование механизма предварительного прореживания рёбер графа позволило повысить качество получаемого разбиения для LSKM на 2 %, BKM на 2,1 %, LSNKM на 3 %, FKM на 3,3 %.

Таким образом, предложенная новая эвристика для PAM-реализации алгоритма k-medoids позволяет существенно снизить вычислительную сложность алгоритма в зависимости от числа вершин и приблизить её к линейной. Основным направлением дальнейшего развития предложенной быстрой модификации алгоритма k-medoids является разработка его параллельной версии для работы на нескольких ядрах или процессорах.

ЛИТЕРАТУРА

1. *Fortunato S.* Community detection in graphs // Physics Reports. 2010. V. 486. Iss. 3–5. P. 75–174.
2. *Kaufman L. and Rousseeuw P. J.* Finding Groups in Data: An Introduction to Cluster Analysis. New Jersey: Hoboken, 2005.
3. *Newman M. J.* Modularity and community structure in networks // Proc. of the National Academy of Sciences of the USA. 2006. V. 103. No. 23. P. 8577–8582.
4. *Satuluri V. M.* Scalable Clustering of Modern Networks. Ohio State University Columbus, OH, USA, 2012.
5. *Ovelgönne M.* Scalable Algorithms for Community Detection in Very Large Graphs. Karlsruhe, 2011. 146 p.
6. `http://snap.stanford.edu/data/` — Stanford Large Network Dataset Collection. 2017.

REFERENCES

1. *Fortunato S.* Community detection in graphs. Physics Reports, 2010, vol. 486, iss. 3–5, pp. 75–174.
2. *Kaufman L. and Rousseeuw P. J.* Finding Groups in Data: An Introduction to Cluster Analysis. New Jersey, Hoboken, 2005.

3. *Newman M. J.* Modularity and community structure in networks. Proc. of the National Academy of Sciences of the USA, 2006, vol. 103, no. 23, pp. 8577–8582.
4. *Satuluri V. M.* Scalable Clustering of Modern Networks. Ohio State University Columbus, OH, USA, 2012.
5. *Ovelgönne M.* Scalable Algorithms for Community Detection in Very Large Graphs. Karlsruhe, 2011. 146 p.
6. `http://snap.stanford.edu/data/` — Stanford Large Network Dataset Collection. 2017.