Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

ЭВРИСТИЧЕСКИЙ АЛГОРИТМ ДЛЯ НАХОЖДЕНИЯ МАКСИМАЛЬНОЙ КЛИКИ В ГРАФЕ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Волкова Даниила Андреевича

Научный руководитель
к. ф.-м. н. _____ С. В. Миронов

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задача о максимальной клике (Maximum Clique Problem, MCP) является одной из самых известных NP-трудных задач в теории графов [1], для которой научное сообщество пока не нашло решения за полиномиальное время.

Кликой в графе называется подмножество вершин, которые образуют полный подграф. Максимальной же кликой в графе называется максимальное по размеру подмножество таких вершин. Что важно, существует обобщенная задача MCP – это задача о клике максимального веса (Maximum Weight Clique Problem, MWCP). В ней каждая вершина графа ассоциируется с положительным целым числом (весом), и задача состоит в нахождении клики с максимальной суммой весов вершин. Несмотря на то, что задачи похожи, они имеют разный практический смысл. Зачастую алгоритмы, которые показали свою эффективность при решении одной задачи, не могут столь же эффективно решать другую.

Для решения обеих задач существуют большое количество алгоритмов. Точные алгоритмы неэффективны при решении NP-полных задач, так как в их основе лежит полный перебор. По этой причине научный интерес составляют эвристические алгоритмы, которые не являются точными, но работают за меньшее время. Все они отличаются точностью, скоростью работы, и зачастую используют специфичные свойства графа. В этой работе будет представлен один из таких алгоритмов.

Практическое применение у задачи о максимальной клики довольно широкое. В биоинформатике MCP используется при компьютерном анализе геномных баз данных, например при поиске потенциальных регуляторных структур рибонуклеиновых кислот. В социальных сетях MCP применяется при кластеризации данных – при разделении различных сообществ на группы (кластеры), обладающие общими свойствами. Выделение кластеров позволяет обрабатывать каждый из них отдельным вспомогательным сервером. В химии задача MCP лежит в основе поиска максимальной общей подструктуры в графе, описывающем структуру химического соединения. Кроме того, MCP является математической моделью ряда задач, возникающих при автоматизации проектирования радиоэлектронной аппаратуры.

В указанных приложениях, как правило, необходимы точные решения для MCP. При этом объем входных данных огромный (входные графы могут

содержать до миллиона вершин). Таким образом, актуальным направлением исследования MCP является разработка новых подходов нахождения точных решений с учетом особенностей графов, возникающих в приложениях.

Целью работы являлось написание эвристического алгоритма, который эффективно решал задачу поиска максимальной клики в графе.

Работа состоит из трех глав: задача о максимальной клике, алгоритм, основанный на доверии к вершинам (TrustCLQ) и эмпирические результаты. В первой главе раскрываются все теоретические данные. В нее включена формулировка задачи, основные понятия, описание существующих алгоритмов решения, возможные методы отсечений. Во второй главе описывается предложенный алгоритм и его отличительные особенности от существующих. В третьей главе приводятся тестовые запуски на графах с известными свойствами и сравнение с другими алгоритмами.

1 Основное содержание работы

Задача о максимальной клике (Maximum Clique Problem, MCP) является одной из самых известных NP-трудных задач в теории графов, для которой научное сообщество пока не нашло решения за полиномиальное время.

Кликой в графе называется подмножество вершин, которые образуют полный подграф. Максимальной же кликой в графе называется максимальное по размеру подмножество таких вершин. Что важно, существует обобщенная задача MCP – это задача о клике максимального веса (Maximum Weight Clique Problem, MWCP). В ней каждая вершина графа ассоциируется с положительным целым числом (весом), и задача состоит в нахождении клики с максимальной суммой весов вершин. Несмотря на то, что задачи похожи, они имеют разный практический смысл. Зачастую алгоритмы, которые показали свою эффективность при решении одной задачи, не могут столь же эффективно решать другую.

Для решения обеих задач существуют большое количество алгоритмов. Точные алгоритмы неэффективны при решении NP-полных задач, так как в их основе лежит полный перебор. По этой причине научный интерес составляют эвристические алгоритмы, которые не являются точными, но работают за меньшее время. Все они отличаются точностью, скоростью работы, и зачастую используют специфичные свойства графа.

Самым первым решением задачи о максимальной клике был алгоритм Брона–Кербоша [2]. Он был опубликован в 1971 голландскими учеными (Coen Bron и Joep Kerbosch). Этот алгоритм является точным и основывается на методе ветвей и границ. Его первая реализация была на языке Algol. В 2006 году японские ученые (Etsuji Tomita, Akira Tanaka, Haruhisa Takahashi) доказали, что алгоритм работает в худшем случае за $O(3^{n/3})$ [3]. Это делает его неприменимым на практике.

Алгоритм Cliquer был опубликован в 2002 году финским ученым (Patric Ostergard) [4]. Алгоритм является самым быстрым среди точных и основывается на методе поиска с возвратом. Кроме MCP он поддерживает графы со взвешенными вершинами и может решать MWCP. Помимо всего он может искать все клики заданного размера. Реализован он на языке C. На данный момент этот алгоритм имеет самую подробную документацию.

Производительность алгоритма Cliquer тесно связана с порядком $v_1, \ldots, v_n$.

Существует несколько идей о том, как упорядочить вершины для повышения производительности. Заметим, что если граф является k–цветным, а вершины упорядочены по классу цветов, то выбор вершины в заданном цветовом классе сразу же исключает из рассмотрения другие этого же цвета. Это является оптимальным упорядочением и улучшается при использовании меньшего количества цветов. Жаль, что вычисление хроматического числа тоже является NP-полной задачей. Но даже жадная неоптимальная раскраска графа способна ускорить алгоритм.

Алгоритм MaxClq был опубликован в 2010 году китайскими учеными (Chu–Min Li и Zhe Quan) [5]. Он является эвристическим и основывается на решении MaxSAT. MaxSAT это задача определения максимального количества клауз булевой формулы в конъюктивной нормальной формуле. Это позволяет эффективно считать верхнюю границу размера клики и отсекать заведомо неудачные решения. Экспериментальные результаты показали высокую эффективность на случайных графах DIMACS.

Способ кодирования MCP в MaxSAT состоит в том, чтобы ввести логическую переменную x для каждой вершины v в G со значением $x = 1$, тогда и только тогда, когда v находится в максимальной клике. Тогда строгая клауза $\overline{x_i} \vee \overline{x_j}$ будет добавлена для каждой пары вершин v_i и v_j, которые не соединены ребром. Это же означает, что они не содержатся в одной клике. Отсюда следует, что каждое присваивание, удовлетворяющее всем строгим клаузам, дает клику. В итоге каждое новое присваивание порождает нестрогую клаузу. Максимизирование количества нестрогих клауз, удовлетворяющих всем строгим, дает максимальную клику.

Например, кодирование графа на рисунке 1 будет выглядеть из одной нестрогой клаузы $\{x_1, x_2, x_3, x_4, x_5, x_6\}$ и нескольких строгих $\{\overline{x_1} \vee \overline{x_4}, \overline{x_1} \vee \overline{x_5}, \overline{x_2} \vee \overline{x_3}, \overline{x_2} \vee \overline{x_5}, \overline{x_3} \vee \overline{x_4}, \overline{x_1} \vee \overline{x_6}, \overline{x_2} \vee \overline{x_6}, \overline{x_4} \vee \overline{x_6}, \overline{x_5} \vee \overline{x_6}\}$

Усовершенствованная версия алгоритма MCS была предложена в 2013 году (Mikhail Batsyn, Boris Goldengorin, Evgeny Maslov, Panos M. Pardalos) [6]. Алгоритм с его усовершенствованиями использует следующие идеи.

Максимальный размер клики произвольного графа не больше его хроматического числа. Это следует из того факта, что клика из k вершин может быть окрашена только в k цветов, потому что его вершины все попарно смежны. Стоит обратить внимание, что хроматическое число графа может быть сколь

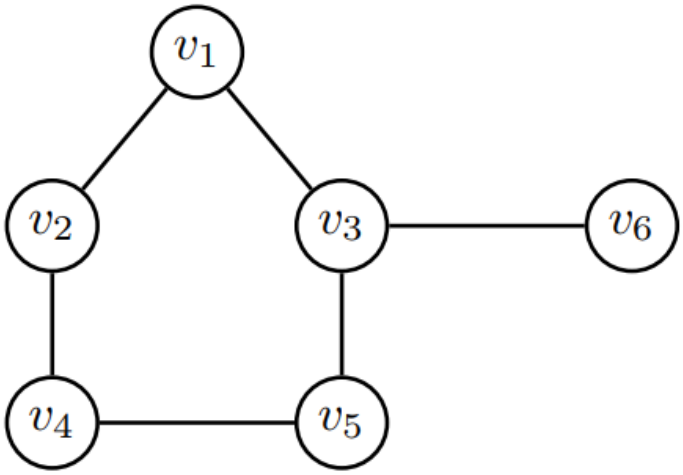

Рисунок 1

угодно большим, чем его клика.

Если граф с k вершинами окрашен в k цветов разумной раскраской, то этот граф полный, т. е. его вершины образуют клику.

Если вершина смежна со всеми вершинами графа, то она будет иметь уникальный цвет в любой раскраске графа.

Предложенный алгоритм был назван TrustCLQ.

Принципиально новым подходом в решении MCP является показатель доверия к вершине. Он имеет интуитивно понятный практический смысл. Рассмотрим следующую формулу: $trust(v) - = \frac{k}{deg(v)+1}$. Если мы ожидаем то, что вершина v сформирует клику размера k, а она этого не делает, то мы теряем к ней доверие. Вершины с большей степенью имеют меньше шансов сформировать с первого раза клику размера k, поэтому к ним доверие тает медленней. Это противоречит утверждению: "Чем выше степень, тем выше шанс сформировать ответ".

Здесь важно заметить, что на самом деле доверие должно зависеть от биномиального коэффициента. Поскольку в графе из n вершин максимум может быть $\frac{n!}{k!(n-k)!}$ клик размера k. Это число неприменимо на практике. По этой причине было решено перейти к сильно упрощенной и приблизительной версии. Она показала отличную точность на разреженных графах.

Стандартное доверие у каждой вершины равно 1. Этот показатель может быть изменен перед запуском алгоритма. Чем выше запас доверия к вершине, тем выше точность ответа. Это видно из того, что запас доверия увеличивает количество итераций, каждая из которых имеет шанс улучшить ответ.

Этот метод позволяет искать клики в графах, о которых совершенно ничего не известно. Здесь не используется понятие раскраски графа и числа его независимых множеств. Такой подход является наиболее общим и универсальным. При этом можно регулировать точность, меняя изначальный запас доверия к вершинам. Если необходимо получить приблизительный ответ на большом графе, то точность можно выставить минимальной. Если же необходим как можно более точный ответ на малом, то точность можно выставить в десятки раз больше.

Рассмотрим еще один важный подход к поиску ответа. Очевидно, что если в графе есть клика размера k, то есть и клика размера $k-1$. Логичным будет то, что не стоит продолжать искать большие клики при отсутствии меньших. Это корректно лишь в точных алгоритмах.

Когда дело касается эвристических алгоритмов, то в дело вступает формула биномиального коэффициента. Из нее можно увидеть, что в определенный момент количество клик убывает. В совокупности с отсечением вершин с плохой степенью это показывает хорошую точность. Таким образом вероятность найти лучший ответ растет.

Удаление лишних вершин всегда действует позитивным образом. Так мы сокращаем размер дерева решений и увеличиваем вероятность нахождения оптимального ответа. Будем рассматривать два вида удалений: перманентное и временное.

Перманентное удаление происходит по критерию степени вершин. Допустим, мы ищем клику размера k в графе. Тогда все вершины с степенью $deg(v)+1<k$ могут быть удалены безвозвратно из графа. Это никоим образом не повлияет негативно на ответ, а только позволит его улучшить за счет того, что плохие вершины не будут влиять на размер дерева решений.

Временное удаление происходит при утрате доверия к вершине. Если доверие вершины v стало в определенный момент меньше нуля $trust(v)<0$, то она может быть временно удалена из графа. Это реализуется использованием локальной копии графа. Такой подход хорошо работает при большом разбросе степеней вершин.

Разработанный алгоритм TrustCLQ состоит из нескольких функций. Основная функция $findMaxClique$ принимает на вход неориентированный невзвешенный граф без мультиребер и петель. Нумерация вершин начинается с нуля.

О верхней границе клики ничего не известно, тогда можно смело предположить, что она равна количеству вершин в графе. О нижней границе тоже ничего не известно, поэтому она равна единице. Ответ изначально является пустым множеством. На выходе алгоритм выдает размер максимальной клики и все вершины, которые входят в нее.

Алгоритм был опробован на нескольких сотнях графов. Все они отличаются по своим характеристикам. Все запуски проводились на процессоре Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz.

Изначально алгоритм был написан для решения MCP на одном единственном графе. Это граф со-упоминаний [7–9]. Он был построен на основе данных поставщиков новостной аналитики с 1 января 2015 года по 22 сентября 2015. Были рассмотрены все новости, выпущенные за этот период. В этом графе вершинами являются компании, а ребрами - факт того, что они оказались вместе в одной новости. Таким образом была получена самая обычная коммуникационная сеть. Она состояла из 7030 вершин и 118890 ребер.

Алгоритм показал в графе клику размера 42 при показателе точности 1000 за 28 секунд.

Туда вошло несколько технологических гигантов (Intel, HP), центров медицинского страхования и IT–компаний. Каждая компания была представлена своей аббревиатурой на бирже. Вот список этих компаний: INTC.O, AET.N, CHS.N, ARMH.O, HCN.N, HPQ.N, BAS.N, BUD.N, BRCM.O, CYBR.O, CSX.N, DE.N, AVGO.O, AMD.O, BJRI.O, COG.N, CONE.O, CTG.O, ENTL.O, EOG.N, FE.N, I.N, EXC.N, COMM.O, DTV.O, ED.N, EQIX.O, HL.N, MO.N, AEP.N, DG.N, EE.N, EIX.N, ELNK.O, IMS.N, OFC.N, ETR.N, FLKS.O, HT.N, HY.N, IBKR.O, KSU.N.

Институтом рисков Саратовского Государственного Университета были взяты 500 крупнейших компаний из вышеупомянутого графа со-упоминаний и построены 72 меньших по размеру за определенный период времени [10–13]. Таким образом каждый граф описывает наличие компаний в одной новости за 1 месяц с января 2005 по декабрь 2010.

Поскольку эти графы сильно разреженные, у них технически возможно посчитать количество максимальных клик. Все результаты исследований были получены с использованием предложенного алгоритма. Для решения задачи о максимальной клике запас доверия вершин был установлен в размере 1000, а

для решения задачи о независимом множестве 1. Все результаты приведены в таблицах 1, 2, 3, 4, 5 и 6.

Таблица 1 – Результаты за 2005 год

Период	# ребер	$\omega(G)$	# $\omega(G)$	$\alpha(G)$
Январь	891	5	9	324
Февраль	955	5	8	327
Март	1091	6	4	317
Апрель	960	5	3	319
Май	1071	5	5	310
Июнь	1090	6	1	314
Июль	911	6	1	313
Август	1036	6	2	308
Сентябрь	1134	7	5	310
Октябрь	1084	5	3	302
Ноябрь	1170	5	13	299
Декабрь	895	5	3	315

Таблица 2 – Результаты за 2006 год

Период	# ребер	$\omega(G)$	# $\omega(G)$	$\alpha(G)$
Январь	994	5	3	313
Февраль	1098	5	8	303
Март	1125	5	8	312
Апрель	908	5	2	312
Май	1070	6	1	298
Июнь	1052	5	3	296
Июль	955	6	1	312
Август	937	5	3	305
Сентябрь	1143	6	2	305
Октябрь	1173	5	3	290
Ноябрь	1158	6	1	292
Декабрь	851	5	1	317

Таблица 3 – Результаты за 2007 год

Период	# ребер	$\omega(G)$	# $\omega(G)$	$\alpha(G)$
Январь	1256	5	9	289
Февраль	1346	6	7	282
Март	1371	7	2	278
Апрель	1291	7	1	288
Май	1444	7	5	279
Июнь	1382	7	2	280
Июль	1334	7	1	277
Август	1168	6	4	293
Сентябрь	1325	6	4	292
Октябрь	1594	7	1	269
Ноябрь	1426	7	5	282
Декабрь	1212	7	3	287

Таблица 4 – Результаты за 2008 год

Период	# ребер	$\omega(G)$	# $\omega(G)$	$\alpha(G)$
Январь	1605	8	1	272
Февраль	1465	7	1	268
Март	1316	8	1	297
Апрель	1238	6	5	289
Май	1249	6	6	293
Июнь	1145	6	2	300
Июль	1312	7	7	289
Август	1003	6	7	313
Сентябрь	1385	10	5	291
Октябрь	1333	7	14	281
Ноябрь	1037	7	2	310
Декабрь	894	5	16	311

Таблица 5 – Результаты за 2009 год

Период	# ребер	$\omega(G)$	# $\omega(G)$	$\alpha(G)$
Январь	1039	6	2	304
Февраль	1122	7	1	301
Март	1090	6	8	295
Апрель	1030	6	8	304
Май	1024	6	4	299
Июнь	1065	5	14	295
Июль	1007	5	8	299
Август	881	5	14	319
Сентябрь	1058	6	3	302
Октябрь	1021	8	1	299
Ноябрь	1006	6	5	299
Декабрь	857	6	6	323

Таблица 6 – Результаты за 2010 год

Период	# ребер	$\omega(G)$	# $\omega(G)$	$\alpha(G)$
Январь	967	6	8	315
Февраль	976	6	2	312
Март	995	6	3	311
Апрель	996	6	3	306
Май	931	8	1	315
Июнь	914	6	10	319
Июль	925	6	2	317
Август	819	6	1	326
Сентябрь	952	5	13	313
Октябрь	941	6	7	317
Ноябрь	946	7	2	314
Декабрь	763	5	6	332

Алгоритм был протестирован на графах научного сообщества DIMACS. В таблице 7 используется стандартная настройка с показателем доверия 1.

Эти результаты наиболее точно описывают эффективность алгоритма. Здесь случайность превращается в закономерность.

Таблица 7 – Результаты тестирования графов DIMACS (показатель доверия равен 1)

Граф	# вершин	# ребер	Известное $\omega(G)$	Найденное $\omega(G)$	Время, с
C125.9	125	6963	34	34	2
C250.9	250	27984	44	40	26
C500.9	500	112332	57	49	710
C1000.9	1000	450079	68	57	15379
C2000.9	2000	1799532	80	-	-
C2000.5	2000	999836	16	15	5744
MANN_a27	378	70551	126	125	224
MANN_a45	1035	533115	345	340	27640
brock200_2	200	9876	12	12	1
brock200_4	200	13089	17	17	3
brock400_2	400	59786	29	24	94
brock400_4	400	59765	33	33	109
brock800_2	800	208166	24	19	940
brock800_4	800	207643	26	26	688
gen200_p0.9_44	200	17910	44	37	12
gen200_p0.9_55	200	17910	55	49	7
gen400_p0.9_65	400	71820	65	48	166
gen400_p0.9_75	400	71820	75	57	204
hamming8-4	256	20864	16	16	6
keller4	171	9435	11	11	1
keller5	776	225990	27	27	1164
p_hat300-1	300	10933	8	8	1
p_hat300-2	300	21928	25	25	4
p_hat300-3	300	33390	36	32	17
p_hat700-1	700	60999	11	11	17
p_hat700-2	700	121728	44	42	209
p_hat700-3	700	183010	62	54	557
p_hat1500-1	1500	284923	12	12	572
p_hat1500-2	1500	568960	65	55	3431
p_hat1500-3	1500	847244	94	76	13035

Алгоритм был запущен так же на настоящих коммуникационных сетях. В таблице 8 можно увидеть примеры запусков. Абсолютно все всех случаях ответ был улучшен.

Таблица 8 – Результаты тестирования графов Facebook (показатель доверия равен 1)

Граф	# вершин	# ребер	Известное $\omega(G)$	Найденное $\omega(G)$	Время, с
socfb-American75	6386	217662	9	40	7
socfb-Amherst41	2235	90954	10	21	6
socfb-Auburn71	18488	973918	13	57	70
socfb-BC17	11509	486967	12	35	31
socfb-BU10	19700	637528	12	38	54
socfb-Baylor93	12803	679817	11	54	41
socfb-Berkeley13	22900	852419	15	42	84
socfb-Bingham82	10004	362894	10	42	15
socfb-Bowdoin47	2252	84387	9	23	4
socfb-Brandeis99	3898	137567	9	23	7
socfb-Brown11	8600	384526	10	34	27
socfb-Bucknell39	3826	158864	9	30	7
socfb-CMU	6621	249959	15	45	8
socfb-Cal65	11247	351358	19	50	13
socfb-Caltech36	769	16656	12	20	1
socfb-Carnegie49	6637	249967	15	45	9
socfb-Colgate88	3482	155043	12	33	7
socfb-Columbia2	11770	444333	10	31	33
socfb-Cornell5	18660	790777	13	40	68
socfb-Dartmouth6	7694	304076	11	28	21
socfb-Emory27	7460	330014	10	52	12
socfb-FSU53	27737	1034802	24	56	132
socfb-GWU54	12193	469528	9	43	26
socfb-Georgetown15	9414	425638	10	33	32
socfb-Hamilton46	2314	96394	15	25	5
socfb-Harward1	15126	824617	9	39	105
socfb-JMU79	14070	485564	9	39	28
socfb-JohnsHopkins55	5180	186586	10	44	5
socfb-Lehigh96	5075	198347	9	37	7
socfb-MIT	6402	251230	9	33	16
socfb-MIT8	6440	251252	9	33	15
socfb-MU78	15436	649449	11	49	36
socfb-Maine59	9069	243247	11	29	11
socfb-Maryland58	20871	744862	9	54	66
socfb-Mich67	3748	81903	14	27	2
socfb-Middlebury45	3075	124610	13	25	8
socfb-Mississippi66	10521	610911	10	48	58
socfb-NYU9	21679	715715	13	37	69
socfb-Northeastern19	13882	381934	11	35	24

ЗАКЛЮЧЕНИЕ

В ходе данной работы был написан эвристический алгоритм для решения задачи поиска максимальной клики. В нем были применены как уже существующие методы отсечения и выбора вершин, так и принципиальные новые, которые ранее в наиболее известных работах не упоминались. Алгоритм использует общий подход к решению MCP, он практически не зависит от порядка вершин. Ему не требуется для работы разбиение графа на независимые множества.

Важным моментом является то, что можно задать необходимую точность алгоритму. Так при показателе доверия равном $\frac{1}{n}$, где n – число вершин в графе, алгоритм имеет минимальную точность. С увеличением этого показателя она растет вместе со временем. Такой подход позволяет проводить как долгие точные, так и быстрые поверхностные исследования. Чем выше показатель доверия, тем меньше алгоритм зависит от того, как поведет себя датчик псевдослучайных чисел.

Алгоритм не требует существенного количества дополнительной памяти для работы. Он хранит только локальную копию графа и несколько не превосходящих количества вершин множеств.

Для времени работы тяжело сформировать оценку, поскольку многое зависит от внутренней структуры графа. В ходе исследования можно было заметить, что большую роль во времени играет среднее значение степени вершины. Несмотря на то, что в социальных графах огромное количество связей, они довольно быстро обрабатываются, так как их плотность невелика.

Алгоритм может применяться и для решения задачи о независимом множестве. В этом случае необходимо взять дополнение исходного графа и подать его на вход программе.

Подсчет количества максимальных клик является в общем случае невыполнимой задачей, но имеет место быть при анализе графов. Данный алгоритм может выполнять и эти вычисления.

Алгоритм был протестирован на огромном количестве различных графов. Он показал высокую точность на графах DIMACS, которые не являются настоящими, но хороши для тестирования алгоритмов решения MCP. Отлично себя показал при запуске на коммуникационных сетях. Полученный ответ максимальной клики превзошел существующие.

Алгоритм был сравнен по времени и точности с одним из самых лучших в мире. Каждый из них хорош в отдельно взятых случаях. Нельзя сказать, что предложенный алгоритм является совершенно новым и может полностью изменить представление решения MCP. Однако он имеет место на существование и в большинстве случаев отлично справляется со своей задачей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Resende, M.* Optimization by GRASP: Greedy Randomized Adaptive Search Procedures / M. Resende, C. Ribeiro. — Springer, 2016.

2 *Bron, C.* Algorithm 457: Finding all cliques of an undirected graph / C. Bron, J. Kerbosch // *Commun. ACM.* — 1973. — Vol. 16, no. 9. — Pp. 575–577.

3 *Tomita, E.* The Worst-case Time Complexity for Generating All Maximal Cliques and Computational Experiments / E. Tomita, A. Tanaka, H. Takahashi. — 2006. — Vol. 363. — Pp. 28–42.

4 Cliquer homepage [Электронный ресурс]. — URL: https://users.aalto.fi/pat/cliquer.html (Дата обращения 03.06.2018).

5 *Li, C.-M.* An efficient branch-and-bound algorithm based on maxsat for the maximum clique problem // Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence. — AAAI'10. — AAAI Press, 2010. — Pp. 128–133.

6 *Batsyn, M.* Improvements to mcs algorithm for the maximum clique problem / M. Batsyn, B. Goldengorin, E. Maslov, P. M. Pardalos // *J. Comb. Optim.* — 2014. — Vol. 27. — Pp. 397–416.

7 *Kim, K.* The determinants of international news flow: A network analysis / K. Kim, G. A. Barnett // *Communication Research.* — 1996. — Vol. 23, no. 3. — Pp. 323–352.

8 *Mangas Vega, A.* Bibliometric study and network analysis of the phenomenon of self publishing // Proceedings of the 3rd International Conference on Technological Ecosystems for Enhancing Multiculturality. — TEEM '15. — New York, NY, USA: ACM, 2015. — Pp. 439–447.

9 *Mitra, G.* The Handbook of News Analytics in Finance / G. Mitra. — John Wiley & Sons, 2011.

10 *Mitra, G.* Handbook of Sentiment Analysis in Finance / G. Mitra. — 2016.

11 *Jeong, H.* The large-scale organization of metabolic networks. / H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, A. L. Barabási // *Nature.* — 2000. — Vol. 407, no. 6804. — Pp. 651–654.

12 *Albert, R.* Statistical mechanics of complex networks / R. Albert, A. Barabási // *Reviews of Modern Physics*. — 2002. — Vol. 74, no. 1. — Pp. 47–97.

13 *Wagner, A.* The small world inside large metabolic networks / A. Wagner, D. A. Fell // *Proceedings of the Royal Society of London B: Biological Sciences*. — 2001. — Vol. 268, no. 1478. — Pp. 1803–1810.