References

1. **Magauenov R. G.** *Sistemy okhrannoi signalizatcii: osnoyy teorii i principy postroeniya* (Guard Signalization Sistems: Base of Theory and Build Principles), Moscow, Goryachaya liniya — Telekom, 2004, 367 p. (in Russian).
2. **Perminov A. N.** *Pravo i bezopasnost'*, 2005, no. 4, pp. 41—46 (in Russian).
3. **Rutkovskaya D., Pilin'skij M., Rutkovskij L.** *Neironye seti, geneticheskie algoritmy i nechiotkie sistemi* (Neuron Networks, Genetic Algorithms and Fuzzy Systems), Moscow, Goryachaya liniya — Telekom, 2006, 452 p. (in Russian).
4. **Holland G.** *V mire nauki,* 1992, no. 9—10, pp. 32 (in Russian).
5. **Borovskij A. S.** *Vestnik komp'uternikh i informatsionnykh tekhnologiy,* 2014, no. 10, pp. 45—52 (in Russian).
6. **Skurikhin A.** *Novosti iskusstvennogo intellekta,* 1995, no. 4, pp. 6—17 (in Russian).
7. **Tarasov A. D., Borovskij A. S.** *Svidetelstvo o gosudarstvennoy registracii programmy dlya EVM* № 2014615742 "GenalgSfz", 02.06.2014. (in Russian).

УДК 519.17, 519.8

А. И. Николаев, стажер-исследователь, ainikolaev@hse.ru,
Лаборатория алгоритмов и технологий анализа сетевых структур,
Национальный исследовательский университет Высшая школа экономики, Нижний Новгород

Эффективный подход на основе машинного обучения к решению задачи о максимальной клике

Представлен новый подход к решению задачи о максимальной клике. Предложенный подход состоит в том, что для данного графа с помощью машинного обучения выбирается наиболее быстрый алгоритм из нескольких алгоритмов, решающих задачу о максимальной клике. После чего выбранный алгоритм применяется для решения задачи о максимальной клике в этом графе. Вычислительные эксперименты на графах библиотеки DIMACS показывают, что представленный подход позволяет с высокой точностью выбрать наиболее быстрый алгоритм из нескольких рассматриваемых.

Ключевые слова: *задача о максимальной клике, точные алгоритмы, деревья классификации*

Введение

Задача о максимальной клике в простом неориентированном графе $G = (V, E)$ является классической задачей теории графов. *Кликой* (или *полным подграфом*) графа G называется такое подмножество его вершин, в котором любые две вершины соединены ребром. Клика, которая не содержится в клике большего размера, называется *максимальной по включению*. Задача о максимальной клике (ЗМК) состоит в том, чтобы для заданного графа G найти клику максимального размера.

ЗМК является *NP*-трудной задачей [4]. Данная задача имеет большое число приложений. В частности, эта задача возникает в теории кодирования [8], биоинформатике [2], анализе социальных сетей, анализе сетей фондовых рынков [1].

Для решения ЗМК в литературе предложено большое число как точных алгоритмов, так и приближенных (эвристик). Подробный обзор точных и приближенных алгоритмов для ЗМК может быть найден в работе [10]. В соответствии с результатами, опубликованными в этом обзоре, алгоритмы MaxCLQ [5] и MCS [9] являются наиболее эффективными современными алгоритмами для точного решения ЗМК.

Так как ЗМК относится к классу *NP*-трудных задач, то теоретический анализ точных алгоритмов для решения ЗМК затруднителен. Вместо этого проводят эмпирические исследования их характеристик на тестовых графах. Например, для тестирования алгоритмов, решающих ЗМК, используют графы библиотеки DIMACS [12]. При этом во многих работах, посвященных данной задаче, делается вывод о том, что новый алгоритм "лучше" других, если он затрачивает на решение всех тестовых графов меньше времени, чем другие алгоритмы. Хотя тот факт, что одни алгоритмы оказываются лучше других алгоритмов на конкретных графах, зачастую никак не учитывается.

Идеальным решением данной проблемы является обращение к некоему "оракулу", который знает, сколько времени тратит каждый из известных алгоритмов на решение данного примера. После чего оракул выбирает алгоритм, который решает рассматриваемую задачу за минимальное время. К сожалению, совершенного оракула, который за короткое время выбирает наиболее быстрый алгоритм, для таких задач не существует, так как задача определения алгоритма является еще более сложной с вычислительной точки зрения. Поэтому на практике мы можем рассматривать только эвристические алгоритмы для предсказания наиболее быстрого алгоритма из предложенного множества алгоритмов.

В настоящей работе представлен подход, который использует машинное обучение для выбора наиболее быстрого алгоритма из нескольких точных алгоритмов для ЗМК, а именно алгоритма MaxCLQ [5], алгоритма MCS [9] и нескольких вариантов алгоритма RPC [7]. С этой целью для ЗМК были выделены признаки, которые используют при работе метода машинного обучения. Для предложенного подхода проведены вычислительные эксперименты, показывающие его эффективность.

1. Задача о максимальной клике

Задача о максимальной клике в графе $G = (V, E)$, где $|V| = n$, может быть сформулирована как задача булева линейного программирования (БЛП):

$$\max \sum_{i=1}^{n} x_i; \tag{1}$$

$$x_i + x_j \leqslant 1, \ \forall (i, j) \in \overline{E} = \\ = \{(i, j) | i, j \in V, \ i \neq j \wedge (i, j) \notin E\}; \tag{2}$$

$$x_i \in \{0, 1\}, \ \forall i = 1, \ ..., \ n. \tag{3}$$

В данной модели если вершина i входит в максимальную клику, то переменная x_i равна 1, в противном случае x_i равна 0. Неравенство (2) гарантирует, что только смежные вершины могут входить в искомое решение.

Так как ЗМК может быть сформулирована как задача БЛП, то ее можно решить с помощью общих методов решения задач БЛП. Однако на практике оказывается, что алгоритмы, специально разработанные для решения ЗМК, требуют гораздо меньше времени на поиск точного решения. Поэтому далее будут рассмотрены три современных алгоритма, разработанных для решения ЗМК.

Алгоритм MCS. Алгоритм MCS был предложен в работе [9]. Этот алгоритм реализует метод ветвей и границ. В качестве верхней границы используется *вершинная раскраска* рассматриваемого графа. Вершинной раскраской графа $G = (V, E)$ называется такое отображение $c : V \to N$, что $c(v) \neq c(w)$ для любых двух смежных вершин $v, w \in V$. Множество вершин, покрашенных в i-й цвет, будем обозначать C_i. Следующее утверждение связывает число вершин в максимальной клике графа G, которое называется *кликовым числом* $\omega(G)$, с числом цветов в вершинной раскраске графа G.

Утверждение 1. Если граф $G = (V, E)$ может быть раскрашен в k цветов, то $\omega(G) \leqslant k$, причем в максимальную клику входит не более одной вершины из каждого цвета.

Отметим, что задача нахождения раскраски с минимальным числом цветов является *NP*-трудной задачей. Поэтому на практике для нахождения вершинной раскраски графа часто используют приближенные алгоритмы, в частности, "жадный" алгоритм. "Жадный" алгоритм раскраски работает следующим образом: алгоритм последовательно красит каждую из вершин, учитывая выбранный порядок вершин, в допустимый цвет с минимальным номером. Для повышения качества "жадной" раскраски в алгоритме MCS используется "перекрашивание" [9].

Алгоритм RPC. Алгоритм RPC (*Reusing Parent Coloring*) был предложен в работе [7]. Алгоритм RPC является параметрическим алгоритмом с целым неотрицательным параметром δ. Алгоритм RPC основан на алгоритме MCS [9], и в нем также для вычисления верхней границы используется "жадная" раскраска с перекрашиванием. Основная идея алгоритма RPC заключается в повторном использовании "родительской" раскраски для некоторых подзадач в дереве поиска вместо трудоемкого вычисления "жадной" раскраски с перекрашиванием для каждой подзадачи. Чем больше значение параметра δ, тем чаще повторно используется раскраска, полученная в родительском узле дерева поиска [7]. Отметим, что при $\delta = 0$ алгоритм RPC совпадает с алгоритмом MCS, так как для всех подзадач для вычисления верхней границы используется "жадная" раскраска с перекрашиванием.

Алгоритм MaxCLQ. Алгоритм MaxCLQ впервые был описан в работе [5]. Данный алгоритм сводит ЗМК к задаче максимальной выполнимости булевой формулы в конъюнктивной нормальной форме. Пусть $x_1, x_2, ..., x_n$ — булевы переменные, принимающие значения 0 (ложь) и 1 (истина). Задача максимальной выполнимости булевой формулы состоит в том, что для данной формулы $F = S_1 \wedge S_2 \wedge ... \wedge S_k$ (где $S_j = \bigvee_{i \in R_j} x_i$, $R_j \subseteq \{1, 2, ..., m\}$, $\forall j = \overline{1,k}$), представленной в конъюнктивной нормальной форме, необходимо определить значения булевых переменных x_i ($i = \overline{1,m}$), при которых максимальное число дизъюнктов S_j будет выполняться (принимать значение 1).

Для сведения ЗМК к задаче максимальной выполнимости алгоритм MaxCLQ каждой вершине i графа G ставит в соответствие булеву переменную x_i. Каждой паре несмежных вершин i, j алгоритм ставит в соответствие дизъюнкт $H_r = \bar{x}_i \vee \bar{x}_j$. Вершинная раскраска $\{C_1, C_2, ..., C_k\}$, полученная с помощью "жадной" раскраски всего графа G, преобразуется в множество дизъюнктов $\{S_1, S_2, ..., S_k\}$ так, что переменная x_i входит в j-й дизъюнкт S_j ($j = \overline{1,k}$) тогда и только тогда, когда $i \in C_j$. Задача о максимальной клике в графе эквивалента задаче максимальной выполнимости булевой формулы $F = S_1 \wedge S_2 \wedge ... \wedge S_k$ с дополнительным ограничением, что все дизъюнкты H_r должны выполняться [6]. В такой постановке значение $x_i = 1$ означает, что вершина i входит в максимальную клику.

2. Новый подход к решению задачи о максимальной клике

2.1 Описание подхода

Предложенный подход к решению задачи о максимальной клике выполняется следующим образом. Сначала для заданного графа G выбирают один из алгоритмов MCS, MaxCLQ и нескольких вариантов алгоритма RPC, определяемых значением параметра δ. Необходимо выбрать тот алгоритм, который с наибольшей вероятностью будет самым быстрым для данного графа. После этого выбранный алгоритм применяют для решения ЗМК для графа G.

Задача выбора (предсказания) наиболее быстрого алгоритма из нескольких была рассмотрена как классическая задача машинного обучения — задача классификации. Для этого были выделены признаки, которые используются при работе метода машинного обучения, сгенерирована обучающая выборка и выбран определенный метод классификации.

2.2 Выделенные признаки для входных данных

В соответствии с результатами, опубликованными в работах по точным алгоритмам для решения ЗМК [5, 7—10], можно сделать вывод о том, что наиболее важными характеристиками графов, влияющими на быстродействие алгоритмов ветвей и границ, являются такие параметры, как размеры и плотность графа, степени вершин и их соседей, нижняя и верхняя границы на размер максимальной клики, ширина дерева поиска. Поэтому для выбора наиболее быстрого точного алгоритма для ЗМК из множества алгоритмов были выделены следующие признаки.

1. Число вершин.
2. Число ребер.
3. Плотность графа.
4. Минимальная степень вершин.
5. Максимальная степень вершин.
6. Среднее значение степени вершины.
7. Среднее квадратичное отклонение степени вершины.
8. Минимальная сумма степеней смежных вершин.
9. Максимальная сумма степеней смежных вершин.
10. Среднее значение суммы степеней смежных вершин.
11. Среднее квадратичное отклонение суммы степеней смежных вершин.
12. Размер клики, найденной "жадной" эвристикой (нижняя граница).
13. Число цветов в "жадной" раскраске графа (верхняя граница).
14. Число вершин, которые нужно рассмотреть на первом уровне дерева поиска (ширина дерева поиска).
15. Доля вершин в графе, которые нужно рассмотреть на первом уровне дерева поиска (относительная ширина дерева поиска).

Признаки 1—3 являются классическими признаками для различных задач на графах. Признаки 4—7 описывают характеристики степеней вершин графа, а признаки 8—11 описывают характеристики суммы степеней смежных вершин. Признак 12 дает нижнюю границу на размер максимальной клики в графе. Для нахождения нижней границы используют два "жадных" алгоритма. Первый "жадный" алгоритм сначала добавляет вершину v_1 с максимальной степенью в текущую клику Q. После этого из вершин, смежных вершине v_1, выбирается вершина v_2 с максимальной степенью в подграфе, образованном этими вершинами, и добавляется в Q. Затем в Q добавляется вершина v_3 с максимальной степенью среди соседей вершин v_1, v_2 и так далее. Алгоритм заканчивает свою работу, когда клика Q является максимальной по включению.

Второй "жадный" алгоритм из всего графа G удаляет вершину с наименьшей степенью и пересчитывает степени всех вершин в новом графе G'. Далее удаляется вершина с наименьшей степенью в графе G' и так далее. Вершины в графе удаляются до тех пор, пока новый граф не станет кликой. Значение признака 12 равно максимуму из размеров клик, полученных двумя эвристиками. Признак 13 дает верхнюю границу на максимальную клику, опираясь на утверждение 1. Вычисление этой верхней границы выполняется с помощью той же "жадной" эвристики для раскраски графа, которая используется в алгоритмах MCS и RPC. Признак 14 показывает число вершин, которые нужно рассмотреть на первом уровне дерева поиска в алгоритмах MCS и RPC при размере текущей максимальной клики $|Q|$, равном значению признака 12. Это все вершины, которые имеют номер цвета в начальной "жадной" раскраске больше $|Q|$, а значит, потенциально могут входить в клику, которая больше, чем Q. Признак 15 показывает долю таких вершин в графе.

2.3 Обучающая выборка

В качестве обучающей выборки использовали графы, сгенерированные в соответствии с моделью Эрдеша—Реньи [3]. Для генерации графов необходимо задать два параметра n и p, где n — число вершин в генерируемом графе, а p — вероятность того, что любые две вершины i и j будут соединены ребром независимо от всех остальных пар вершин. В табл. 1 представлены значения параметров сгенерированных графов. Вероятности p задавались с шагом 0,05. Для каждой пары значений n и p было сгенерировано 10 графов.

Алгоритмы MCS, RPC и MaxCLQ были реализованы на языке C++ и запущены на 950 сгенерированных графах. Реализация алгоритма MaxCLQ была любезно предоставлена его авторами. Все вычисления проводили на персональном компьютере со следующими характеристиками: процессор Intel Core i5 2,3 ГГц, размер оперативной памяти 8 Гбайт.

Таблица 1

Параметры сгенерированных графов

n	p
150	0,05—0,95
200	0,05—0,95
300	0,05—0,85
400	0,05—0,6
500	0,05—0,6
1000	0,05—0,4
1500	0,05—0,4

Время работы каждого алгоритма было ограничено 600 с. Как результат, только 947 графов были решены за отведенное время. Стоит отметить, что алгоритм RPC запускался с различными параметрами δ начиная с $\delta = 1$. После того как алгоритм RPC закончил свою работу с параметром $\delta = 1$, запускался алгоритм RPC с параметром $\delta = 2$ и так далее. В случае если алгоритм RPC с новым значением параметра $\delta = k$ тратил больше времени, чем алгоритм RPC с параметром $\delta = k - 1$, то алгоритм останавливался и увеличение параметра δ прекращалось. Таким образом, в результате решения ЗМК для всех сгенерированных графов было получено, что $1 \leqslant \delta \leqslant 3$.

2.4. Метод классификации

Наиболее быстрый алгоритм выбирали из следующих алгоритмов: MaxCLQ, MCS и RPC со значениями параметра $\delta = 1, 2, 3$. На рисунке (см. третью сторону обложки) представлено распределение графов из обучающей выборки на классы. Здесь 1 — это алгоритм MaxCLQ, который оказался наиболее быстрым для 83 графов; 2 — алгоритм MCS (наиболее быстрый для 320 графов); 3 — алгоритм RPC с параметром $\delta = 1$ (348 графов); 4 — RPC с $\delta = 2$ (188 графов); 5 — RPC с $\delta = 3$ (8 графов). В случае произвольного графа (после вычисления всех его признаков) задачу определения наиболее быстрого алгоритма решения ЗМК для этого графа можно рассматривать как задачу классификации, где номер класса совпадает с номером наиболее быстрого алгоритма.

Задача классификации является известной и широко изучаемой задачей машинного обучения. Для ее решения существует большое число методов, например, метод опорных векторов, байесовский классификатор, нейронная сеть и др. В данной работе был использован метод "деревья классификации" [6], так как этот метод быстро работает и возвращает результаты, которые легко интерпретировать и анализировать. Также несомненным достоинством этого метода является отбор информативных признаков при обучении классификатора. Стоит отметить, что деревья классификации используются для бинарной классификации, но могут быть применены для многоклассовой классификации. Так, в данной работе использована стратегия "один против всех" для многоклассовой классификации. Стратегия "один против всех" состоит в том, что вместо одного классификатора обучается M классификаторов, где M — общее число классов. Классификатор i (в нашем случае i-е дерево классификации) отделяет объекты i-го класса от объектов остальных классов. Результатом работы дерева классификации является вероятность $P(i \mid x)$ того, что объект x относится к i-му классу. Затем для объекта x выбирается класс j^* с максимальной вероятностью: $j^* = \arg \max\limits_{j = \overline{1, M}} P(j \mid x)$.

В нашем случае M равно 5, поэтому с помощью программного обеспечения RapidMiner [13] было обучено пять классификаторов. Стоит отметить, что полученные в результате обучения деревья классификации использовали не все признаки, а только следующие: 2, 3, 6, 8, 10, 12, 13, 14. Это означает, что остальные признаки либо оказались несущественными, либо зависимыми от выбранных признаков. Таким образом, вычисление этих признаков не требуется.

3. Вычислительные результаты

Предложенный алгоритм был обучен на случайных графах, описанных в подразд. 2.2, и протестирован на графах библиотеки DIMACS.

Из библиотеки DIMACS было рассмотрено 25 графов (см. столбец 1 в табл. 3), для которых рассматриваемые алгоритмы решают ЗМК за время, превышающее 1 с (чтобы исключить погрешности компьютерных вычислений) и не превышающее 2 ч (чтобы сократить общее время вычислений). Стоит отметить, что для вычисления признаков, описанных в подразд. 2.1, для любого из графов DIMACS было потрачено не более 20 мс. Результаты работы метода предсказания наиболее быстрого алгоритма для графов библиотеки DIMACS представлены в табл. 2. Для 13 графов наиболее быстрый — алгоритм 4, предсказанный верно для всех 13 графов. Для восьми графов наиболее быстрым является алгоритм 5, но был предсказан алгоритм 4, который, тем не менее, дает близкие результаты и для всех восьми графов является вторым по скорости вы-

Таблица 2

Таблица сопряженности классификатора для графов DIMACS

		Реальный класс					Точность, %
		1	3	4	5	2	
Предсказанный класс	1	2	1	0	0	0	66,67
	3	0	1	0	0	0	100,00
	4	0	0	13	8	0	61,90
	5	0	0	0	0	0	0,00
	2	0	0	0	0	0	0,00
	Полнота, %	100,00	50,00	100,00	0,00	0,00	

Таблица 3

Время решения ЗМК для графов библиотеки DIMACS, мс

Граф DIMACS	MaxCLQ	MCS	RPC, δ = 1	RPC, δ = 2	RPC, δ = 3	Наиболее быстрый алгоритм	Предложенный подход
C250.9	344516	1361335	1041168	987836	971229	344516	344517
MANN_a45	34148	43230	31159	34574	85558	31159	34154
brock400_1	259708	284586	244209	234715	235103	234715	234716
brock400_2	118908	125398	107746	103844	103952	103844	103845
brock400_3	204222	193386	164518	158995	159121	158995	158996
brock400_4	130754	92893	79692	76856	77071	76856	76858
brock800_1	5606592	3914626	3430660	3294158	3300223	3294158	3294165
brock800_2	4889039	3395345	2971610	2845765	2842208	2842208	2845772
brock800_3	3222601	3461488	3020965	2899545	2898460	2898460	2899552
brock800_4	2438408	1602096	1405760	1341436	1333342	1333342	1341442
dsjc500.5	3532	1555	1412	1426	1487	1412	1415
dsjc1000.5	317877	140509	127070	123663	127732	123663	123675
frb30-15-1	655244	721617	440375	339713	289979	289979	339715
frb30-15-2	951654	533285	296789	208696	160926	160926	208698
frb30-15-3	580959	473354	278528	210579	176525	176525	210581
frb30-15-4	1155555	1327562	765939	574914	481628	481628	574915
frb30-15-5	873662	489109	266500	187442	147959	147959	187443
p_hat300-3	1387	1245	1051	1044	1089	1044	1045
p_hat500-3	49829	60698	51642	49310	49779	49310	49313
p_hat700-2	3586	2726	2319	2250	2371	2250	2255
p_hat700-3	1082242	1063763	894519	841329	854369	841329	841333
p_hat1000-2	117828	106392	87986	82428	83927	82428	82439
p_hat1500-1	11408	2257	1993	1916	1922	1916	1936
sanr200_0.9	5604	13855	10712	10107	10122	5604	5604
sanr400_0.7	97663	79299	68261	66440	68368	66440	66442
Общее время	23156926	19491609	15792583	14678981	14464450	13750666	14030826

числений (табл. 3). Для двух графов наилучший алгоритм — алгоритм 1, и он был предсказан верно в обоих случаях. И еще для двух графов наилучшим является алгоритм 3, вместо которого в одном случае был предсказан алгоритм 1, дающий второй по скорости вычислений результат, близкий к первому. Общая точность предсказания алгоритма равна 64 % (16 из 25).

В табл. 4 показано среднее сокращение времени вычислений с помощью предложенного подхода (подсчет признаков, выбор одного из пяти алгоритмов на основе деревьев классификации, решение ЗМК с помощью выбранного алгоритма) по сравнению с каждым из рассматриваемых алгоритмов в отдельности. Среднее сокращение времени вычислений относительно каждого отдельного алгоритма подсчитывается следующим образом: для каждого графа вычисляется, насколько время работы предложенного подхода меньше времени работы отдельного алгоритма, а затем вычисляется среднее арифметическое по всем полученным значениям. Сокращение времени вычислений предложенного подхода определяется по формуле $\frac{t' - t}{t'} \cdot 100\ \%$, где t — время работы предложенного подхода, а t' — время работы отдельного алгоритма. Опираясь на табл. 3 и 4, можно сделать вывод о том, что предложенный подход в среднем быстрее алгоритма MaxCLQ и алгоритма MCS на 35 и 28 % соответственно. Если же сравнивать предложенный подход с заранее неизвестным наиболее быстрым для каждого конкретного графа алгоритмом, то наиболее быстрый алгоритм (разный для каждого графа) тратит на решение ЗМК лишь на 4,97 % времени меньше, чем предложенный подход.

Таблица 4

Среднее сокращение времени вычислений (%) предложенного подхода относительно каждого из рассматриваемых алгоритмов в отдельности для графов библиотеки DIMACS

MaxCLQ	MCS	RPC, δ = 1	RPC, δ = 2	RPC, δ = 3	Наиболее быстрый алгоритм	Предложенный подход
35,74	28,97	11,83	4,41	3,21	-4,97	0,00

Заключение

В данной работе представлен новый подход к решению задачи о максимальной клике. Предложенный подход использует деревья классификации для выбора из нескольких рассматриваемых алгоритмов наиболее быстрого алгоритма для заданного графа. После этого выбранный алгоритм применяется для решения задачи о максимальной клике

в данном графе. С этой целью для ЗМК выделены признаки, которые используют при предсказании наиболее быстрого алгоритма. Вычислительные результаты показывают эффективность предложенного подхода. Так, для графов библиотеки DIMACS точность выбора наиболее быстрого алгоритма составляет 64 %, а в тех случаях, когда выбранный алгоритм не является наиболее быстрым, он является вторым по скорости вычислений. При этом наиболее быстрый алгоритм (разный для каждого графа) тратит на решение ЗМК лишь на 5 % меньше, чем предложенный подход.

Работа поддержана грантом РНФ 14-41-00039.

Список литературы

1. **Boginski V., Butenko S., Pardalos P.** Statistical analysis of financial networks // Computational Statistics and Data Analysis. 2005. N. 48 (2). P. 431—443

2. **Butenko S., Wilhelm W. E.** Clique-detection models in computational biochemistry and genormics // European Journal of Operational Researc. 2006. N. 173. P. 1—17.

3. **Erdos P., Renyi A.** On random graphs I // Publ. Math. Debrecen. 1959. N. 6. P. 290—297.

4. **Karp R. M.** Reducibility Among Combinatorial Problems // Complexity of Computer Computations. 1972. P. 85—103.

5. **Li C. M., Quan Z.** Combining graph structure exploitation and propositional reasoning for the maximum clique problem // Proceedings of the 2010 22nd IEEE International Conference on Tools with Artificial Intelligence. 2010. Vol. 1. P. 344—351.

6. **Loh W.-Y.** Classification and regression trees // Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2011. Vol. 1, N. 1. P. 14—23.

7. **Nikolaev A., Batsyn M., San Segundo P.** Reusing the same coloring in the child nodes of the search tree for the maximum clique problem // Lecture Notes in Computer Science. 2015. N. 8994. P. 275—280.

8. **Pardalos P., Rebennack S.** Computational challenges with cliques, quasi-cliques and clique partitions in graphs // Lecture Notes in Computer Science. 2010. N. 6049. P. 13—22.

9. **Tomita E., Sutani Y., Higashi T., Takahashi S., Wakatsuki M.** A Simple and Faster Branch-and-Bound Algorithm for Finding a Maximum Clique // Lecture Notes in Computer Science. 2010. N. 5942. P. 191—203.

10. **Wu Q., Hao J. K.** A review on algorithms for maximum clique problems // European Journal of Operational Research. 2015. N. 242. P. 693—709.

11. **Xu L., Hutter F., Hoos H. H., Leyton-Brown K.** SATzilla: portfolio-based algorithm selection for SAT // Journal of Artificial Intelligence Research, 2008, vol. 32, pp. 565—606.

12. **http:**//dimacs.rutgers.edu/Challenges/

13. **http:**//rapidminer.com/

A. I. Nikolaev, Research Intern, ainikolaev@hse.ru
Laboratory of Algorithms and Technologies for Networks Analysis,
National Research University Higher School of Economics, Nizhny Novgorod

Efficient Approach for the Maximum Clique Problem Based on Machine Learning

In this paper a new approach for solving the maximum clique problem is presented. For a given graph the suggested approach uses machine learning technique to predict the fastest algorithm from several algorithms for the maximum clique problem. Then the chosen algorithm is applied for solving the maximum clique problem in this graph. The computational results show the efficiency of the proposed approach.

Keywords: *maximum clique problem; exact algorithms; decision trees*

References

1. **Boginski V., Butenko S., Pardalos P.** Statistical analysis of financial networks, *Computational Statistics and Data Analysis*, 2005, no. 48 (2), pp. 431—443.

2. **Butenko S., Wilhelm W. E.** Clique-detection models in computational biochemistry andgenomics, *European Journal of Operational Research*, 2006, 173, pp. 1—17.

3. **Erdos P., Renyi A.** On random graphs I, *Publ. Math. Debrecen*, 1959, no. 6, pp. 290—297.

4. **Karp R. M.** Reducibility Among Combinatorial Problems, *Complexity of Computer Computations*, 1972, pp. 85—103.

5. **Li C. M., Quan Z.** Combining graph structure exploitation and propositional reasoning for the maximum clique problem, *Proceedings of the 2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, 2010, no. 1, pp. 344—351.

6. **Loh W.-Y.** Classification and regression trees, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2011, vol. 1, no. 1, pp. 14—23.

7. **Nikolaev A., Batsyn M., San Segundo P.** Reusing the same coloring in the child nodes of the search tree for the maximum cliqueproblem, *Lecture Notes in Computer Science*, 2015, no. 8994, pp. 275—280.

8. **Pardalos P., Rebennack S.** Computational challenges with cliques, quasi-cliques and clique partitions in graphs, *Lecture Notes in Computer Science*, 2010, 6049, pp. 13—22.

9. **Tomita E., Sutani Y., Higashi T., Takahashi S., Wakatsuki M.** A Simple and Faster Branch-and-Bomd Algorithm for Finding a Maximum Cliquey, *Lecture Notes in Computer Science*, 2010, no. 45942, pp. 191—203.

10. **Wu Q., Hao J. K.** A review on algorithms for maximum clique problems, *European Journal of Operational Researches*, 2015, no. 242, p. 693—709.

11. **Xu L., Hutter F., Hoos H. H., Leyton-Brown K.** SATzilla: portfolio-based algorithm selection for SAT, *Journal of Artificial Intelligence Research*, 2008, vol. 32, pp. 565—606.

12. **http:**//dimacs.rutgers.edu/Challenges/

13. **http:**//rapidminer.com/