obj2node

Матвеев Артем и Никишкина Евгения, 317

1. Основная постановка

Дано: число $s \in \mathbb{N}$ и набор чисел $\{m_{ij}\}_{i<j}, m_{ij} \in \mathbb{N} \cup \{0\}$

Построить: множество M минимальной мощности, удовлетворяющее следующим ограничениям:

1. среди всех подмножеств M имеются s подмножеств $M_1, \ldots, M_s$
2. для каждой пары из $\frac{s(s-1)}{2}$ пар (M_i, M_j) данных подмножеств известна мощность их пересечения $|M_i \cap M_j| = m_{ij}$

Решением задачи является множество M с указанием s его подмножеств.

Определение. Набор $(M, M_1, \ldots, M_s)$, удовлетворяющий ограничениям 1 и 2, будем называть *решением* поставленной задачи.

Определение. Набор $(M, M_1, \ldots, M_s)$, в котором M имеет наименьшую мощность среди всех решений поставленной задачи, будем называть *оптимальным решением* поставленной задачи.

1.1 Существование решения для основной постановки

Утверждение. $\forall s \in \mathbb{N}$ и для любого набора $\{m_{ij}\}$ существует решение задачи мощности $\sum_{i<j} m_{ij}$.

Доказательство. Будем строить множество M конструктивно на основе следующего алгоритма:

Algorithm 1 Построение множества M и набора подмножеств $M_1, \ldots, M_s$

1: $k = 1, M_1 = \varnothing, \ldots, M_s = \varnothing$
2: **Пока** во множестве $\{m_{ij}\}$ есть ненулевые элементы:
3: Выбираем любой ненулевой элемент m_{ij} и $M_i = M_i \cup \{k\}, M_j = M_j \cup \{k\}$
4: $k = k + 1$
5: $m_{ij} = m_{ij} - 1$
6: **Итог:** $M = \{1, \ldots, \sum_{i<j} m_{ij}\}, M_1, \ldots, M_s$

$k \in M_i$ и $k \in M_j \iff k$ был добавлен на одной из интераций цикла. Таких добавлений будет ровно $m_{ij} \Rightarrow |M_i \cap M_j| = m_{ij} \Rightarrow (M, M_1, \ldots, M_s)$ решение поставленной задачи. $\square$

1.2 Критерии качества решения для основной постановки

Было показано, что решение поставленной задачи всегда существует, но как будет показано далее, поиск оптимального решение - это сложная задача, поэтому введем метрики качетсва полученного решения:

1. Мощность множетсва M. Ее мы хотим минимизировать.
2. Доля пар (M_i, M_j), для которых выполнено $|M_i \cap M_j| = m_{ij}$. Хотим максимизировать.

По этим метрикам будет производиться сравнение методов решения задачи, которые будут представлены далее.

1.3 Эквивалентная постановка

Дано: число $s \in \mathbb{N}$ и набор числе $\{m_{ij}\}_{i<j}, m_{ij} \in \mathbb{N} \cup \{0\}$

Рассмотрим неориентированный взвешенный граф $G = (V, E)$ такой, что:

1. $|V| = s$, каждой вершине взаимно однозначно поставлено число из множества $\{1, \ldots, s\}$
2. $|E| =$ числу ненулевых m_{ij}

Ребро (i, j, w) принадлежит множеству ребер графа G, если во множестве $\{m_{ij}\}$ существует ненулевой элемент m_{ij}, причем $w = m_{ij}$.

Построить: множество M, состоящее из клик, M минимальной мощности, удовлетворяющее следующим ограничениям:

1. $\forall$ clique $= (V_c, E_c) \in M \Rightarrow V_c \subset V, E_c \subset E$
2. $\forall$ clique $= (V_c, E_c) \in M, \forall\ (i, j, w) \in E_c \Rightarrow w = 1$
3. Любое ребро (i, j, m_{ij}) графа G покрыто ровно m_{ij} кликами из множества M

Решением задачи является множество M.

Определение. Множество M, удовлетворяющий ограничениям 1, 2 и 3, будем называть *решением* поставленной задачи.

Определение. Множество M, которое имеет наименьшую мощность среди всех решений поставленной задачи, будем называть *оптимальным решением* поставленной задачи.

Утверждение. Постановки 1 и 2 эквивалентны.
Доказательство. Очевидно следует из построения графа G.

1.4 Критерии качества решения для эквивалентной задачи

1. Мощность множетсва M. Ее мы хотим минимизировать.
2. Доля ребер (i, j, m_{ij}) в графе G, для которых выполнено, что ребро покрывается в точности m_{ij} кликами. Хотим максимизировать.

2. NP-полнота задачи

НУЖНА ПОСТАНОВКА ЗАДАЧИ РАСПОЗНОВАНИЯ Очевидно можно показать, что задача не проще, чем NP-полная, сведением частного случая к задаче ECC-problems (edge clique covering problem). ECC - NP-полная (Garey Johnson (1979), Problem GT59). $\Rightarrow$ поставленная задача не проще, чем NP-полная

Доказать, что она NP-полная!

3. Приближенные методы решения

3.1 Описание исходных данных

Данные представляют из себя натуральное число s и множество неотрицательных целых чисел $\{m_{ij}\}_{1\leq i<j\leq s}$. Исходные данные были сгенерированы следующим образом:

1. Была выбрана группа студентов 2-го курса ВМК МГУ из социальной сети "Вконтакте".
2. Далее было зафиксировано множество всех групп, в которые входит хотя бы один из выбранных студентов. Мощность этого множества групп обозначим s.
3. Для фиксированного набора пользователей и набора групп обозначим число $m_{ij}, 1 \leq i, j \leq s$ как мощность множества студентов, которые входят как в группу i, так и в группу j.

Таким образом были получены следующие два набора данных:

1. Мощность множества зафиксированных групп $s = 91$, мощность множества зафиксированных студентов 115(т.е. известно решение мощности 115) и множество чисел $\{m_{ij}\}_{1\leq i<j\leq s}$, построенных по зафиксированному набору групп и набору студентов.
2. Мощность множества зафиксированных групп $s = 878$, мощность множества зафиксированных студентов 2312(т.е. известно решение мощности 2312) и множество чисел $\{m_{ij}\}_{1\leq i<j\leq s}$, построенных по зафиксированному набору групп и набору студентов.

3.2 Релаксация задачи

Для того, чтобы перейти к задаче релаксации, немного изменим исходную постановку задачи. Пусть теперь нужно строить не множество M, имеющее наименьшую мощность и удовлетворяющее всем ограничениям, а множество M, мощность которого меньше или равна некоторого заранее заданного числа C. Тогда эту задачу можно переформулировать в терминах матриц следующим образом:

Дано: Матрица $J \in \mathbb{R}^{s\times s}$ такая, что $J_{ij} = J_{ji} = m_{ij}, 1 \leq i < j \leq s,\ J_{ii} = 0$.

Построить: Матрицу $G \in \{0,1\}^{s\times C}$ такую, что

$$GG^T \circ P = J$$

где $P = \begin{bmatrix} 0\,1\,1\,\ldots\,1 \\ 1\,0\,1\,\ldots\,1 \\ \ldots \\ 1\,1\,1\,\ldots\,0 \end{bmatrix}$

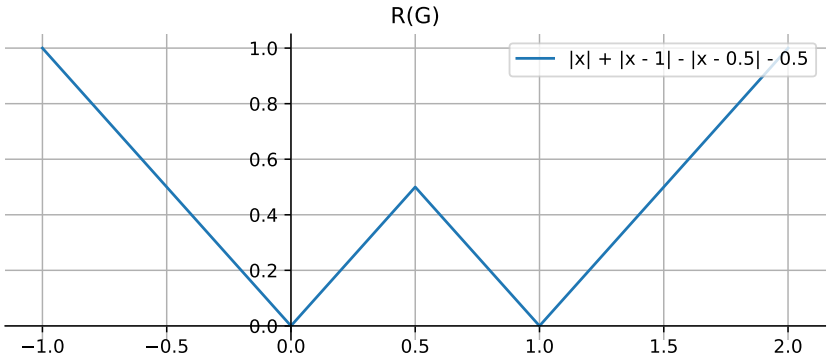
В данном случае роль элементов множества M выполняют столбцы матрицы G, а $G_{ij} = 1$ означает, что множеству i принадлежит элемент с номером j. Соответсвенно, скалярное произведение двух строк есть мощность пересечения соответсвующих множеств.

Перейдем от дискретных ограничений к непрерывным, т.е. $G \in \mathbb{R}^{s \times C}$. Т.к. не можем напрямую оптимизировать долю пар множеств (M_i, M_j), для которых выполнено $|M_i \cap M_j| = m_{ij}$, то будем минимизировать следующий целевой функционал:

$$L(G) = \text{MAE}(GG^T \circ P, J) + \alpha R(G) \to \min_G$$

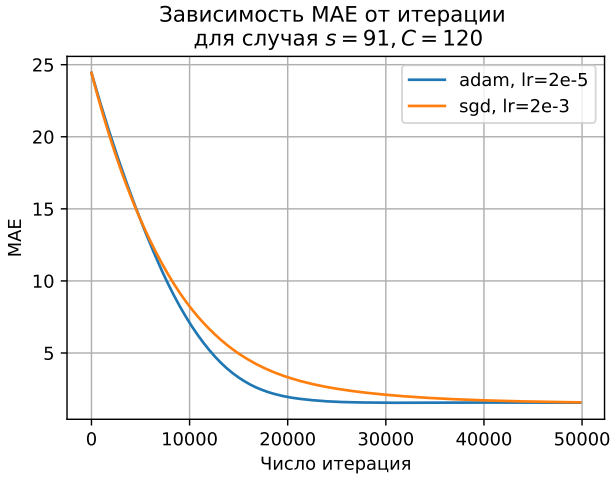
где $R(G)$ - регуляризатор, который штрафует за отклонение перемернных от 0 и 1.

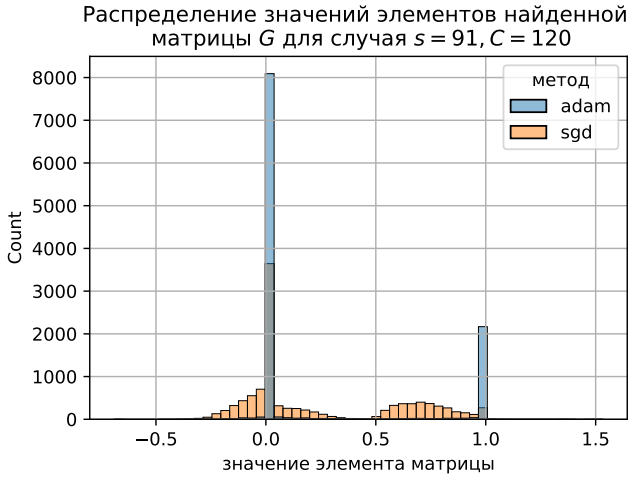
В качестве регуляризатора $R(G)$ возьмем $R(G) = |x| + |x-1| - |x-0.5| - 0.5$.



Задача минимизации будет решаться для двух полученных наборов данных: с $s = 91$ и с $s = 878$. Темпы обучения и значение параметра α были подобраны в ручную.

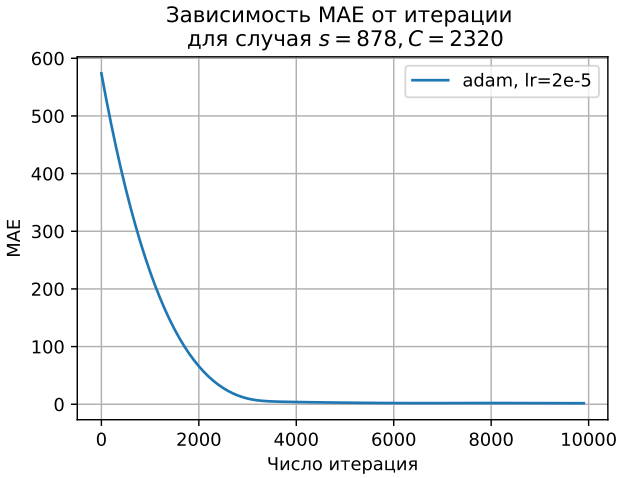
Для набора данных с $s = 91$ известно, что существует решение мощности 115, поэтому будем искать решение, мощность которго не провосходит 120: $C = 120$. Параметр $\alpha = 20$.

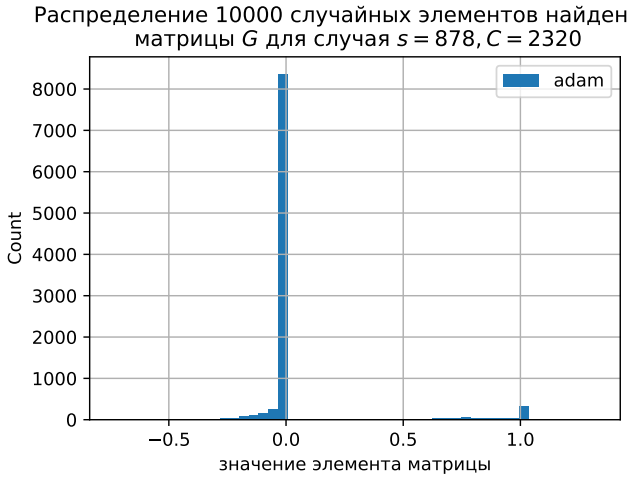




Как Adam, так и SGD после 50000 итераций застряли в локальном минимуме нашего целевого функционала $F(G)$. При этом Adam получил значение MAE 1.55, а SGD 1.58. Заметим, что решение, которое получил Adam почти полностью состоит из 0 и 1, в то время как для SGD это не выполняется, поэтому для набора данных с $s = 878$ будем использовать только Adam.

Для набора данных с $S = 878$ известно, что существует решение мощности 2312, поэтому будем искать решение, мощность которго не превосходит 2320: $C = 2320$. Параметр $\alpha = 40$.





Аналогично после 10000 итераций Adam застрял в локальном минимуме целевого функционала $F(G)$. При этом значение MAE находится в окрестности 1.8. Решение почти полностью состоит из 0 и 1.

Т.е. ограничения на мощность пересечения множеств в обоих случаях точно выполнить не получилось.

3.3 Жадный алгоритм, включающий поиск максимальных клик

Т.к. решение задачи релаксации не выполняет ограничения, наложенные на решение задачи(подмножества должны иметь заданные пересечения), то предложим следующий алгоритм, который гарантировано будет эти ограничения соблюдать.

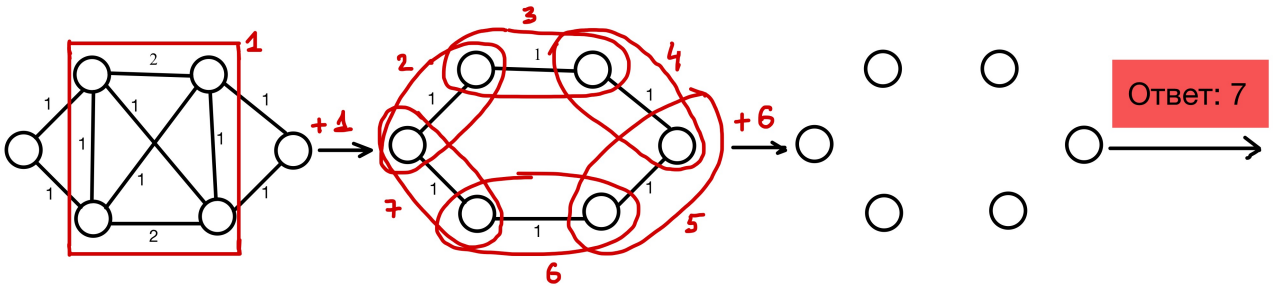
Рассмотрим эквивалентную графовую постановку задачи. Ранее мы показали, что ее решение эквивалентно решению исходной задачи. Будем строить множество M, состоящее из клик, следующим образом:

1. Выбираем наибольшую по числу вершин клику в графе.
2. Уменьшаем на единицу веса всех ребер исходного графа, входящих в эту клику. Если вес стал равен нулю, то удаляем это ребро из исходного графа.
3. Если в графе не осталось ребер, то завершаем алгоритм и получившийся набор клик будет решением, иначе переходим к шагу 1.

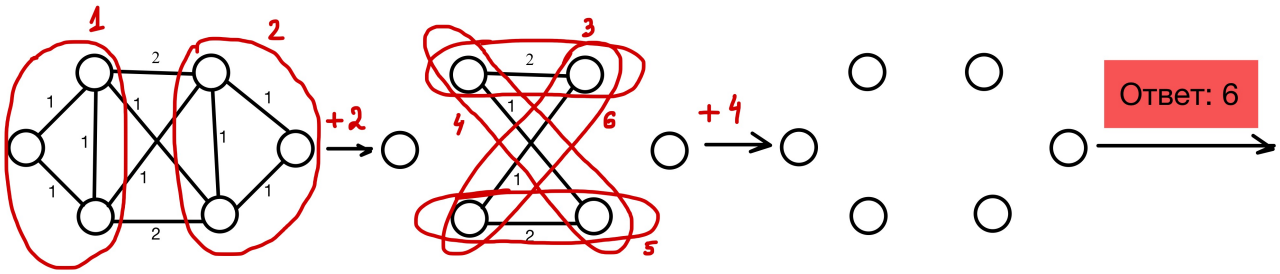
Решение, полученное алгоритмом, действительно является решением задачи, т.к. по построению любое ребро будет покрываться в точности таким числом клик, как его вес, т.е. ограничения будут выполняться.

Интуиция данного жадного алгоритма заключается в том, что так как мы хотим уменьшить число клик в множестве M, то логично будет на каждом шаге выбирать клику в графе, имеющую наибольшее число вершин.

Данный алгоритм действительно является жадным и далеко не всегда находит оптимальное решение. Рассмотрим его работу на графе ниже.



Алгоритм получает множество клик мощности 7, хотя есть очевидное решение мощности 6:



Возникает еще одна проблема. Поиск максимальной клики в графе является NP-полной задачей. Т.к. поиск максимальной клики вызывается на каждой итерации предложенного жадного алгоритм, то эту максимальную клику нужно искать быстро, поэтому точные алгоритмы поиска максимальной клики здесь не применимы. Тогда воспользуемся приближенными алгоритмами поиска.

Выбор этого приближенного алгоритма был сделан на основе статьи 2015 года "A review on algorithms for maximum clique problems. Qinghua Wu, Jin-Kao Hao".

Важно, чтобы алгоритм поиска максимальной клики был быстрым, т.к. он запускается на каждой итерации предложенного жадного алгоритма. При этом алгоритм должен выдавать результаты, которые являются одними из лучших на сегодняший день. На эту роль был выбран алгоритм DLS. В статье этот алгоритм описывается как один из наиболее эффектинвых алгоритмов, показывающих высококонкурентные результаты с рядом алгоритмов, существовавших до DLS. Алгоритм основан на идеи динамического локального поиска, описанного в статье "Dynamic Local Search for the Maximum Clique Problem. Wayne Pullan, Holger H. Hoos.".

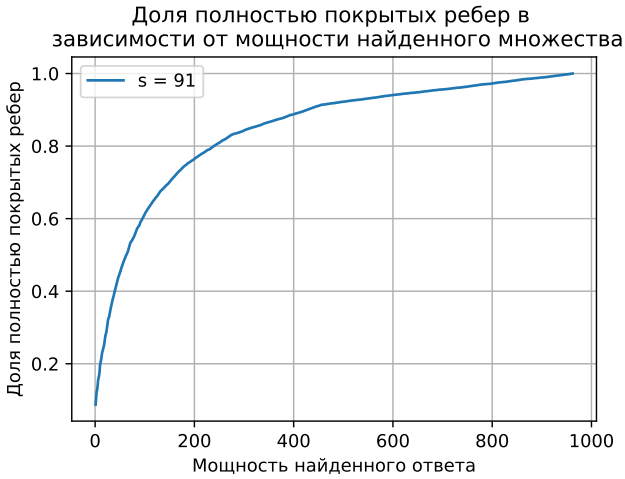
Результаты 21 эффективного эфристического алгоритма по поиску максимальной клики на 9 сложных графах из DIMACS, “–” означает, что результат неизвестен. Средний размер клики находится в круглых скобках, если 100 процентный результат не был достигнут. Среднее время, требуемое на каждый алгоритм находится в колонке Time. Алгоритмы, которые сравниваются между собой: RLS (Battiti Protasi, 2001), GENE (Marchiori, 2002), DAGS (Grosso et al., 2004), VNS (Hansen et al., 2004), KLS (Katayama et al., 2005), EA/G (Zhang et

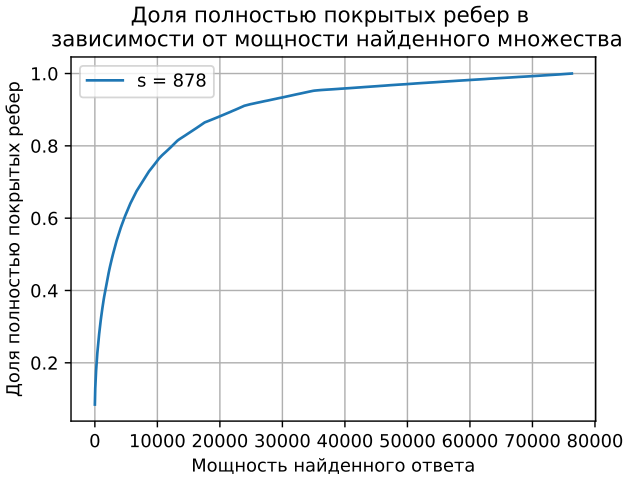
al., 2005), QUALEX-MS (Busygin, 2006), DLS (Pullan Hoos, 2006), PLS (Pullan, 2006), HSSGA (Singh Gupta, 2006a), COVER (Richter et al., 2007), IEA-PTS (Guturu Dantu, 2008), CLS (Pullan et al., 2011), EWCC (Cai et al., 2011), MN/TS (Wu et al., 2012), ILS (Andrade et al., 2012), AMTS (Wu Hao, 2011), BLS (Benlic Hao, 2013) and SBTS (Jin Hao, 2015).

Instance	brock400_2		brock400_4		brock800_2		brock800_4		C2000.9	
	Best(Avg)	*Time*	*Best(Avg)*	*Time*	*Best(Avg)*	*Time*	*Best(Avg)*	*Time*	*Best(Avg)*	*Time*
RLS	29(26.06)	3.06	33(32.42)	7.89	21	0.34	21	0.48	78(77.58)	59.83
GENE	24(22.5)	0.27	25(23.6)	0.19	20(19.3)	0.75	20(18.9)	1.12	72(68.2)	4.89
DAGS	29(28.10)	0.62	33	0.62	24(20.82)	3.73	26(22.60)	3.75	76(75.40)	405.33
VNS	29(27.4)	4.17	33	2.69	21	0.85	21	3.16	78(77.2)	22.74
KLS	25(24.84)	0.04	25	0.01	21(20.86)	0.16	21(20.67)	0.39	77(74.90)	4.80
EA/G	25(24.7)	1.42	33(25.1)	1.42	21(20.1)	3.42	21(19.9)	3.42	72(70.9)	17.38
QUALEX-MS	29	0.67	33	0.44	24	4.00	26	4.00	72	47.78
DLS	29	0.12	33	0.02	24	3.97	26	2.24	78(77.93)	48.79
PLS	29	0.12	33	0.02	24	4.08	26	2.30	78	50.11
HSSGA	29(25.1)	0.14	33(27.0)	0.29	21(20.7)	1.35	21(20.1)	0.38	74(71.0)	14.83
COVER	28(–)	–	33(–)	42.63	–	–	–	–	78(77.84)	139.36
IEA-PTS	29(27.52)	1.08	33	0.84	24(21.06)	1.03	26(21.4)	2.07	79(76.4)	19.71
CLS	29	0.08	33	0.02	24	1.73	26	0.58	78	7.28
EWCC	29(25.48)	374.52	33	25.37	21	0.49	21	0.62	79(78.56)	858.73
MN/TS	29	0.81	33	0.17	24(23.88)	94.25	26	37.57	80(78.37)	339.57
ILS	25	11.57	33(30.3)	11.57	21	63.15	26(21.3)	63.15	77(76.9)	108.42
AMTS	29	0.69	33	0.35	24	19.61	26	9.01	80(78.95)	266.33
BLS	29	10.29	33	1.87	24(23.04)	637.94	26	356.05	80(78.6)	2846.84
SBTS	29	11.97	33	0.49	24(22.29)	464.12	26(25.90)	249.47	80(77.29)	896.78

Instance	C4000.5		MANN_a45		MANN_a81		keller6	
	Best(Avg)	*Time*	*Best(Avg)*	*Time*	*Best(Avg)*	*Time*	*Best(Avg)*	*Time*
RLS	18	158.65	345(343.60)	28.98	1098	205.72	59	13.79
GENE	16(15.4)	1.95	343(342.4)	19.56	1097(1096.3)	401.41	55(51.8)	8.71
DAGS	18(17.50)	717.55	344(343.95)	426.95	–	–	57(56.40)	2739.11
VNS	18	310.71	345(344.5)	1.51	1100(1099.3)	65.47	59(58.2)	17.90
KLS	18(17.02)	7.76	345(343.88)	2.02	1100(1098.07)	12.88	57(55.59)	17.08
EA/G	17(16.1)	23.46	345(343.7)	30.84	1098(1097.2)	319.04	56(53.4)	24.26
QUALEX-MS	17	521.11	342	3.78	1096	106.00	53	286.89
DLS	18	45.76	344	13.12	1098(1097.96)	66.66	59	43.05
PLS	18	47.01	344	84.27	1098	172.17	59(57.75)	172.17
HSSGA	17(16.8)	19.97	343(342.6)	8.22	1095(1094.2)	503.99	57(54.2)	39.67
COVER	18	260.27	345(344.41)	–	1100(–)	–	59	5.89
IEA-PTS	18(17.66)	104.21	345(343.97)	9.43	1099(1097.01)	237.55	59(57.06)	45.40
CLS	18	13.63	344	15.40	1098	20.80	59	0.91
EWCC	18	738.91	345(344.94)	698.50	1100(1098.11)	634.46	59	3.76
MN/TS	18	86.96	340	54.56	1090	380.87	59	58.96
ILS	18(17.1)	1996.84	345(344.5)	3.15	1100	10.52	59	546.31
AMTS	18	74.93	345(344.04)	66.77	1098	16.30	59	6.39
BLS	18	387.33	342(340.82)	–	1094(1092.17)	–	59	14.67
SBTS	18	919.07	345	16.30	1100	13.43	59	446.67

Алгоритм DLS почти на каждом из рассматирваемых тестовых графах выдает наилучший результат, при этом работает быстро и прост в реализации, поэтому выбор пал на него.





До полного выполнения ограничей на покрытие ребер кликами предложенный алгоритм сделал **962** итерации для первого набора данных и **76304** итерации для второго набора данных. Этот результат сильно превосходит известные нам ответы в 115 и 2312 соответсвенно, зато в отличии от подхода с сведением к релаксационной задаче дает ответ, который гарантировано удовлетворяет всем ограничениям.

У предложенного итерационного алгоритма есть хорошее свойство: на ранних итерациях доля полностью покрытых ребер кликами растет очень быстро, а на поздних итерациях растет очень медленно. Тогда если немного изменить постановку задачи добавивив, что допустимым является любое решения, доля полностью покрытых ребер в котором больше или равна некоторого фиксированного числа, то и ответы будут получаться более приемлемые.

В нашем случае, если в качестве такого порога взять число 0.8(т.е. нам подходят все решения, которые правильно восстанавливают 80 процентов всех пересечений множеств), то предложенный алгоритм найдет множества мощности 200 и 10000, что уже значительно больше похоже на известные мощности в 115 и 2312.