Компьютерная математика

Теория и методы оптимизации

Предложен приближенный алгоритм решения задачи нахождения максимального независимого множества вершин графа. С помощью этого алгоритма улучшено известное рекордное значение мощности максимального независимого множества для одного из графов.

УДК 519.854

И.П. ГРАДИНАР

ПРИБЛИЖЕННЫЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ НАХОЖДЕНИЯ МАКСИМАЛЬНОГО НЕЗАВИСИМОГО МНОЖЕСТВА ВЕРШИН ГРАФА

Введение. Известно, что задача нахождения максимального независимого множества вершин графа имеет многочисленные практические приложения [1–3]. Для ее решения в данной работе предложен и исследован приближенный алгоритм. Использование этого алгоритма позволило найти известные из литературы рекордные значения мощности независимого множества, а для одного графа улучшить его.

Постановка задачи. Пусть задан неориентированный граф $G = (V, E)$, где $V = \{v_1, \ldots, v_k\}$ – множество его вершин, $E = \{e_1, \ldots, e_m\}$ – множество ребер ($k = |V|$, $m = |E|$), $|V|$ – мощность множества V.

Приведем необходимые определения. Множество $I \subset V$ вершин графа G называется независимым, если никакие две его вершины не связаны ребром, т. е.

$$I = \{v \in V \mid \forall v_1, v_2 \in I, \exists e = (v_1, v_2) \in E\}.$$

Независимое множество I_{max} называется максимальным независимым множеством, если для любого независимого множества I выполняется соотношение $|I_{max}| \geq |I|$. Обозначим $\alpha(G)$ число элементов в I_{max}, т. е. $\alpha(G) = |I_{max}|$. Независимое множество I называется максимальным по включению, если оно не является подмножеством некоторого другого независимого множества.

Известно [1], что задача отыскания максимального независимого множества вершин графа сводится к задаче целочисленного программирования с булевыми переменными вида

$$\max\left\{ f(x)=\sum_{i=1}^{k} x_i \prod_{j:(v_i,v_j)\in E} (1-x_j) : x\in B^k \right\},$$

где каждой вершине $v_j \in V$ графа $G=(V,E)$ ставится в соответствие переменная $x_j \in \{0,1\}$, $j=1,\ldots,k$.

Пусть

$$I(x)=\{v_j \in V \mid x_j=1,\ j=1,\ldots,k\}, \text{ где } x=(x_1,\ldots,x_k).$$

Если выполняется условие $f(x)=|I(x)|$, то $I(x)$ – независимое множество вершин графа $G=(V,E)$.

Поскольку рассматриваемая задача – NP-трудная [1], то актуальным является вопрос разработки приближенных алгоритмов ее решения. На сегодняшний день известен ряд алгоритмов [1, 2, 4–7] для этой задачи. Среди них алгоритмы, использующие локальный поиск: вектора спада, табу, глобального равновесного поиска и др. Каждый из них имеет свои особенности и преимущества при применении к поставленной задаче. Обзор некоторых приближенных алгоритмов решения данной задачи приведен в [3].

Описание предлагаемого алгоритма. Пусть $N_G(v)$ – множество вершин неориентированного графа $G=(V,E)$, инцидентных вершине $v\in V$, т. е.

$$N_G(v)=\{u\in V \mid \exists e=(u,v)\in E\},$$

и пусть для $V'\subset V$

$$N_G(V')=\bigcup_{\forall v\in V'} N_G(v).$$

Обозначим $deg_G(v)$ степень вершины v (количество вершин, инцидентных вершине v), т.е. $deg_G(v)=|N_G(v)|$. Пусть $\delta(G)$ – число, равное наименьшей степени вершин графа $G=(V,E)$, т.е. $\delta(G)=\min\{deg_G(v) \mid \forall v\in V\}$.

Предположим, что $G[V']=(V',E\cap(V'\times V'))$ – подграф графа $G=(V,E)$, порожденный подмножеством $V'\subset V$.

Предлагаемый в данной работе алгоритм заключается в следующем.

Если v принадлежит независимому множеству I, то никакая вершина из $N_G(v)$ не может быть включена в множество I. Множество вершин, которые не могут быть включены в I, обозначим T, т.е. $T=N_G(I)$.

При формировании множества I есть вершины, не принадлежащие множеству T, и еще не включенные в I, т.е. они не связаны в графе G ни с одной вершиной из множества I и есть претендентами на включение в независимое множество. Множество таких вершин обозначим

$$A=V\setminus I\setminus N_G(I) \text{ или } A=\{v\in V \mid v\notin I;\forall u\in I,\exists e=(v,u)\in E\}.$$

Множества I, A и T определены таким образом, что они являются разбиениями множества V, т.е. $V = I \cup A \cup T$, а их попарное пересечение равняется пустому множеству. Таким образом, каждая вершина графа G обязательно должна принадлежать только одному из этих трех множеств.

В зависимости от того, к какому из множеств I, A или T принадлежит вершина $v \in V$, вершины v' из окрестности $N_G(v)$ могут принадлежать таким множествам:

1) $\forall v \in I, \forall v' \in N_G(v), \quad v' \in T$;

2) $\forall v \in A, \forall v' \in N_G(v), (v' \in A) \vee (v' \in T)$;

3) $\forall v \in T, \forall v' \in N_G(v), (v' \in A) \vee (v' \in T)$.

Очевидно, что если $A = V$, то $I = \varnothing$, $T = \varnothing$. Если же $A = \varnothing$, то это значит, что I – максимальное по включению независимое множество, возможно (но не обязательно), максимальное независимое множество.

При включении в I некоторой вершины v, а это возможно, когда она принадлежит A, из последнего множества она удаляется. Те вершины из окрестности $N_G(v)$, которые принадлежат множеству A (т.е. $N_G(v) \cap A$ или $N_{G[A]}(v)$), переходят в множество T. Другие вершины не изменяют своего "статуса". Схематично это выглядит следующим образом:

$$I \xleftarrow{v} A \xrightarrow{N_G(v) \cap A} T.$$

При удалении из I определенной вершины v она автоматически переходит в множество A. Вершины из окрестности $N_G(v)$ (все они принадлежат T при $v \in I$), которые не имеют связей ни с одной из вершин из I, кроме v, переходят в множество A. Другие вершины не изменяют своей принадлежности к одному из соответствующих множеств I, A или T. Схематично это имеет вид

$$I \xrightarrow{v} A \xleftarrow{N_G(v) \setminus N_G(I \setminus v)} T.$$

Предположим, что рассмотренные особенности включения в I и удаления из I вершины v с соответствующими последствиями для окрестности $N_G(v)$ полностью реализовываются следующими базовыми процедурами:

```
procedure включение_в_I _вершины (номер_вершины_из_A)
  v ← A[номер_вершины_из_A]
  удаление из A вершины v и включение ее в множество I
  удаление из A вершин, принадлежащих N_G(v), и включение их в T.

procedure удаление_из_I _вершины (номер_вершины_из_I)
  v ← I [номер_вершины_из_I]
  удаление из I вершины v и включение ее в множество A
  удаление из T вершин v'∈ N_G(v)\N_G(I \ v) и включение их в A.
```

В дальнейшем будем считать, что доступ к множествам I, A и T для внесения изменений осуществляется только с помощью этих двух процедур. Без этих процедур возможно лишь считывание информации о состоянии этих множеств, а именно об их мощности и содержимом.

Обозначим $A^{\delta} = \{v \in A \mid deg_{G[A]}(v) = \delta(G[A])\}$ подмножество множества вершин A, имеющих наименьшую степень $\delta(G[A])$ в графе $G[A]$. С помощью следующей процедуры постепенно и рандомизированно удаляются из множества A вершины, принадлежащие $A^{\delta} \subset A$, они затем включаются в I. Это происходит до тех пор, пока множество A не станет пустым.

```
procedure рандомизированное_включение_в_ I _вершин_из_ A^δ ()
    while A≠∅ do
        включение_в_ I _вершины(номер в A вершины A^δ [random(| A^δ |)])
```

Приведем процедуру, которая рандомизированно удаляет из I определенное количество вершин.

```
procedure рандомизированное_удаление_из_ I _вершин (кол-во_на_удаление)
    j :=0, количество_на_удаление:=min{ кол-во_на_удаление, | I |}
    while j < количество_на_удаление do
        удаление_из_ I _вершины(random(| I |))
        j := j +1
```

Вышеописанные базовые процедуры будут использованы далее. Схему предложенного приближенного алгоритма представим в виде таких процедур:

```
procedure начальная_настройка (выделенное_время)
    конец:=false
    время_окончания_работы_алгоритма:=now+ выделенное _время
    количество_стартов:=0

procedure новое_начальное_решение ()
    I := ∅ , A := V , T := ∅
    рандомизированное_включение_в_ I _вершин_из_ A^δ ()
    количество_стартов:=количество_стартов+1
    средняя_начальная_мощность_ I := (средняя_начальная_мощность_ I ·
                    ·(количество_стартов–1)+| I | ) / количество_стартов
    увеличилась_мощность_ I ()
    i_max :=random([|V |*0.03])+[|V |*0.001]+1
```

```
procedure увеличилась_мощность_ I ()
    наибольшая_мощность_ I :=| I |
    граница_спада_мощности_ I := | I | – 20
    i_del :=1
    i :=0

procedure приближенный_алгоритм(выделенное_время, ожидаемый_результат)
    начальная_настройка(выделенное_время), количество_стартов:=0
    while (not конец) do
        новое_начальное_решение()
        if | I |≥ожидаемый_результат then конец:=true
        while (| I |> средняя_начальная_мощность_I)and
                                        and ( i_del <2/3·| I |)and(not конец) do
            while ( i < i_max )and(not конец) do
                рандомизированное_удаление_из_ I _вершин( i_del )
                рандомизированное_включение_в_ I _вершин_из_ A^δ ()
                if наибольшая_мощность_ I <| I | then
                    увеличилась_мощность_ I ()
                    if | I |≥ожидаемый_результат then конец:=true
                i := i +1
            i_del := i_del +1
            i :=0
            if | I |< граница_спада_мощности_ I  then
                i_del :=1
                граница_спада_мощности_ I := граница_спада_мощности_ I  – 20
            if now<время_окончания_работы_алгоритма then конец:=true
```

Суть алгоритма заключается в попытке заменить в текущем множестве I i_del вершин большим количеством вершин и тем самым увеличить мощность этого множества. Благодаря рандомизированному выбору среди вершин наименьшей степени, которые появились в множестве A после удаления i_del вершин из множества I, мощность I не уменьшается или существенно не уменьшается. Но когда за количество шагов $i\_\max$ не увеличивается мощность I, необходимо увеличить i_del. Если увеличивается $|I|$, то i_del принимает значение 1, и снова формируем множество I большей мощности при малых значениях i_del. Когда i_del довольно большое, то возможно существенное уменьшение мощности независимого множества и нет смысла дальше продолжать поиск.

В таком случае поиск необходимо завершить и продолжить работу алгоритма с нового начального решения. В процессе работы алгоритма не сохраняются независимые множества для дальнейшего использования, кроме текущего и наибольшего независимых множеств, т. е. алгоритм не нуждается в большом объеме памяти.

Результаты экспериментальных расчетов. Данный алгоритм был применен для ряда графов. Программная реализация предложенного алгоритма была скомпилирована с помощью Borland Delphi 5.0 (build 5.62) и запущена на компьютере с конфигурацией Intel® Core™2 CPU, T7200 @ 2.00GHz, 2GB RAM при загрузке 50 %. Каждая задача решалась 10 раз с ограничением по времени в 5 часов. Полученные результаты представлены в таблице. В колонке 5 этой таблицы приведено среднее время получения наибольшей мощности независимого множества по десяти запускам, в колонке 6 среднеквадратичное отклонение времени получения наибольшей мощности. В колонке 7 приведены из [2] полученные результаты нахождения максимального независимого множества и затраченное на них время (« – » означает, что граф не рассматривался, «*» – что рекорд не достигнут).

Из результатов, приведенных в таблице, видно, что получены известные рекордные значения [1, 4, 5, 8] мощности независимого множества для всех графов. Кроме графа $1zc8192$, рекордные значения найдены в результате всех 10 запусков алгоритма. Для графа $1zc8192$ с помощью предложенного алгоритма улучшено известное [1, 6] рекордное значение от значения 701 до 704. Значение 704 для этого графа получено в результате 9 из 10 решений задачи, при одном решении получено 703. При разработке алгоритма, который предлагается в данной работе, внимание было нацелено на 1zc -графы, для одного из которых и улучшено известное рекордное значение. Для оценки эффективности алгоритма поставленная задача была решена и для DIMACS-графов. Сравнив результат нахождения рекордов по этим графам с результатами, полученными алгоритмом VSA в работе [2] (таблица), можно сказать, что предложенный алгоритм проявляет приемлемую эффективность и для других графов.

Графики (рис. 1–5) схематично иллюстрируют работу предложенного алгоритма. На них отображены:

1) моменты увеличения i_del (рис. 1–3);

2) изменение $|I|$ в зависимости от значения i_del (рис. 2–5);

3) момент, когда $|I|$ увеличивается, а i_del принимает значение 1 (рис. 2, 3);

4) случаи нерационального увеличения i_del или сильного спада мощности I, когда процесс поиска начинается из нового решения (рис. 4, 5);

5) спад $|I|$ к значению граница_спада_мощности_I, когда i_del присваивается 1, а переменной граница_спада_мощности_I присваивается меньшее значение (рис. 5).

ТАБЛИЦА. Результаты вычислительных экспериментов

Название графа	Известное рекордное значение мощности	Полученное наилучшее значение мощности	Среднее значение мощно-сти	Среднее время, *c*	Отклоне-ние, *c*	Результат и время в [2], *c*
1	2	3	4	5	6	7
keller4.clq	11	11	11	0,000	0,000	11, <1
johnson8-4-4.clq	14	14	14	0,000	0,000	14, <1
hamming8-4.clq	16	16	16	0,000	0,000	16, 2
san200_0.7_2.clq	18	18	18	1,381	1,242	18, <1
hamming10-4.clq	40	40	40	0,191	0,161	40, 23
MANN_a27.clq	126	126	126	0,014	0,009	125, *
hamming8-2.clq	128	128	128	0, 000	0,000	128, 3
hamming10-2.clq	512	512	512	0,002	0,005	512, 9
MANN_a45.clq	345	345	345	1118,198	555,325	343 , *
1zc1024	112	112	112	193,908	182,502	– , –
1zc2048	198	198	198	507,191	320,477	– , –
1zc4096	379	379	379	1880,859	1896,986	– , –
1zc8192	701	704	703,9	9977,937	7098,894	– , –

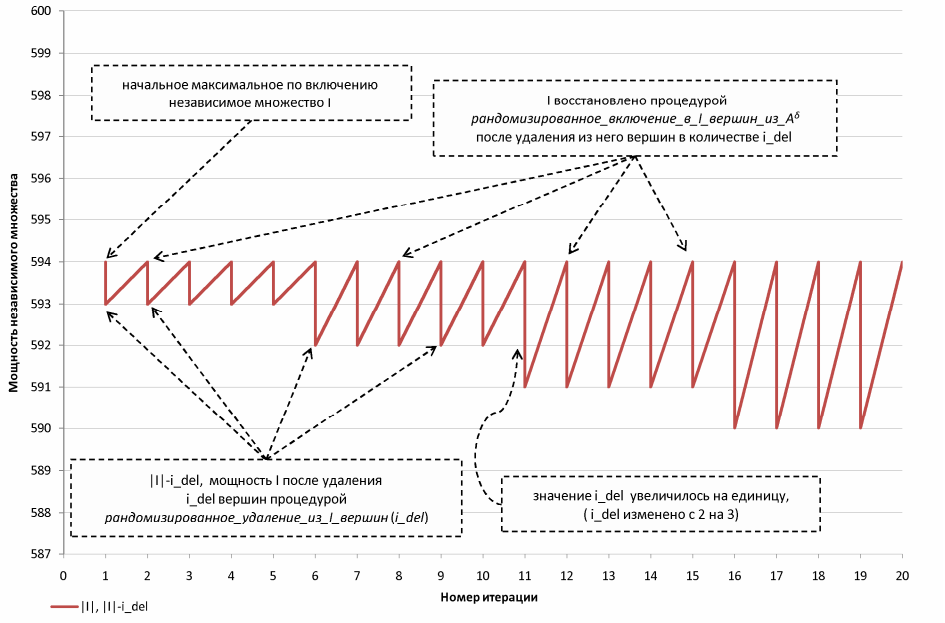

РИС. 1

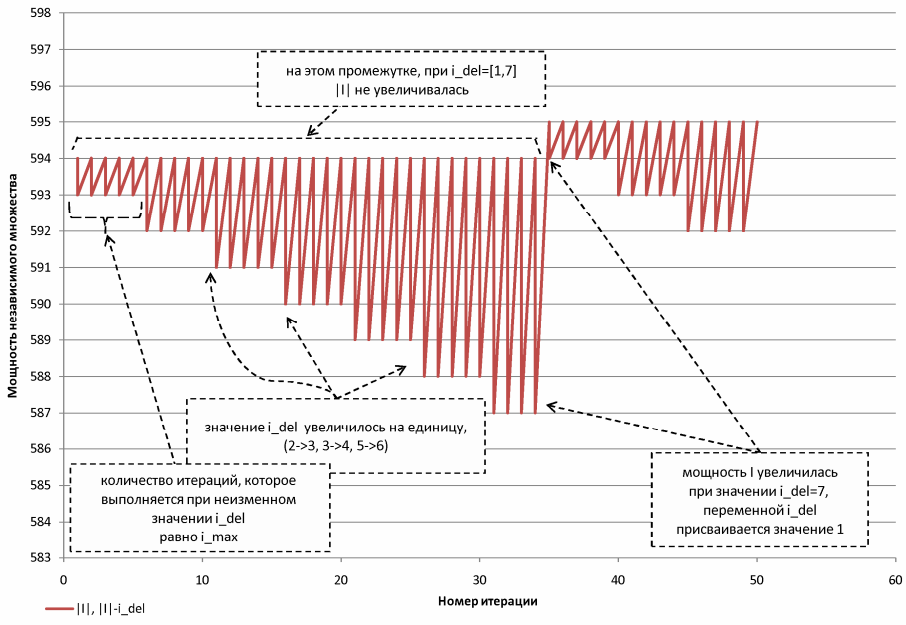
Мощность независимого множества
Номер итерации
|I|, |I|-i_del
на этом промежутке, при i_del=[1,7]
|I| не увеличивалась
значение i_del увеличилось на единицу,
(2->3, 3->4, 5->6)
количество итераций, которое
выполняется при неизменном
значении i_del
равно i_max
мощность I увеличилась
при значении i_del=7,
переменной i_del
присваивается значение 1

РИС. 2

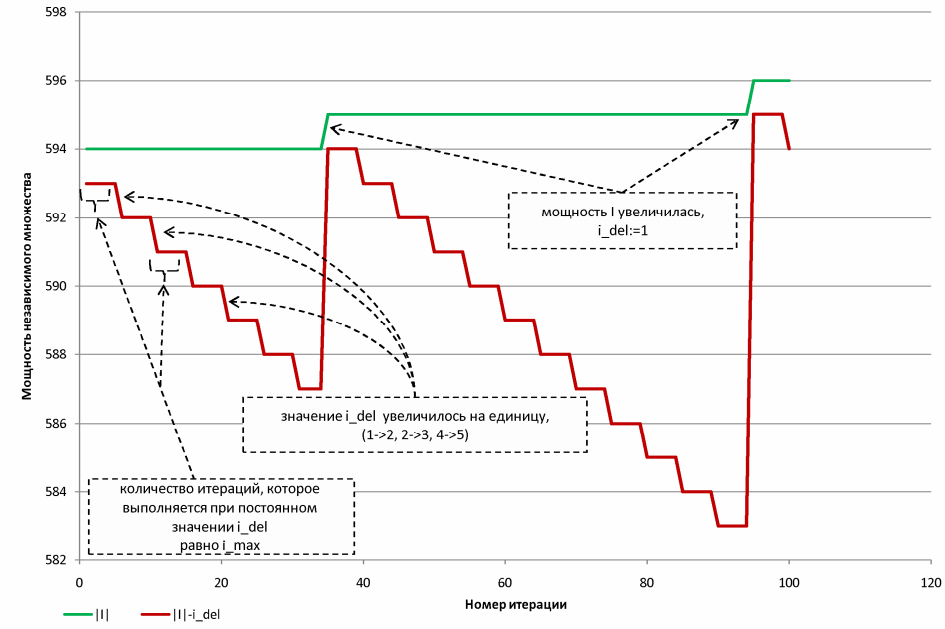
Мощность независимого множества
Номер итерации
|I|
|I|-i_del
мощность I увеличилась,
i_del:=1
значение i_del увеличилось на единицу,
(1->2, 2->3, 4->5)
количество итераций, которое
выполняется при постоянном
значении i_del
равно i_max

РИС. 3

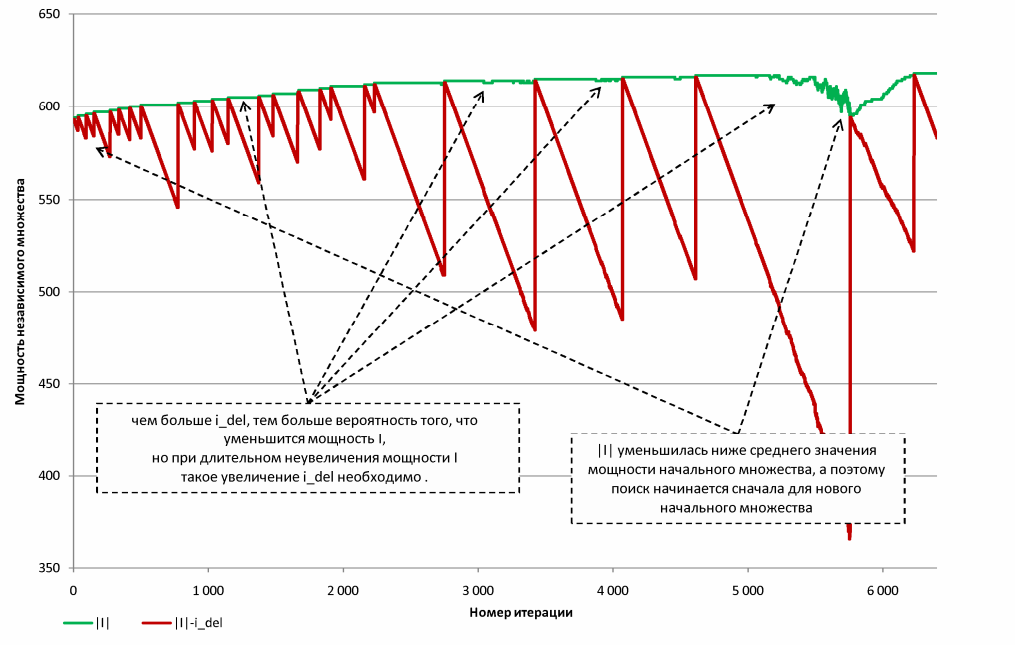
650
600
550
500
450
400
350
Мощность независимого множества
0
1 000
2 000
3 000
4 000
5 000
6 000
Номер итерации
|I|
|I|-i_del
чем больше i_del, тем больше вероятность того, что
уменьшится мощность I,
но при длительном неувеличения мощности I
такое увеличение i_del необходимо .
|I| уменьшилась ниже среднего значения
мощности начального множества, а поэтому
поиск начинается сначала для нового
начального множества

РИС. 4

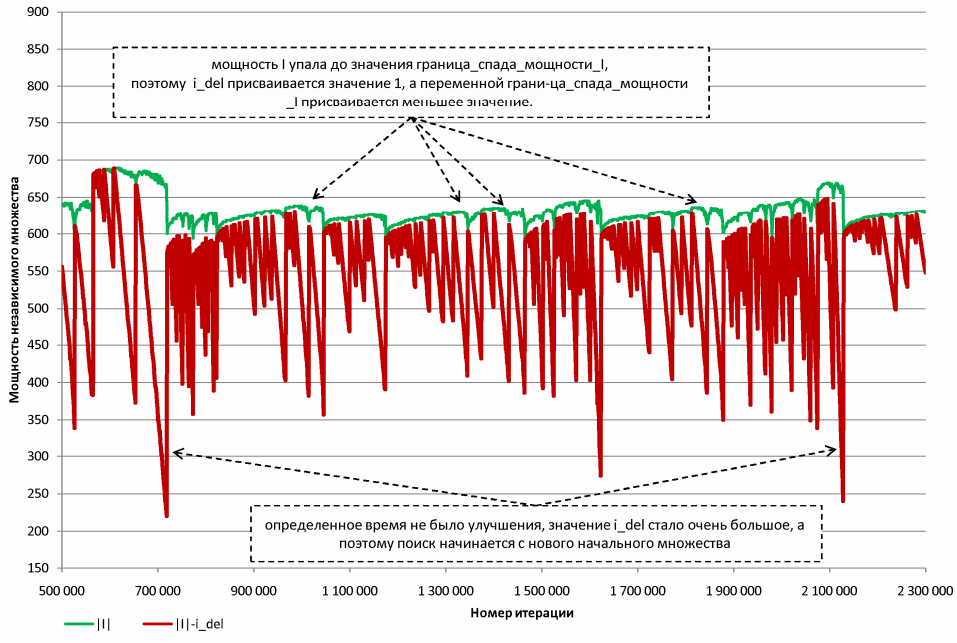
900
850
800
750
700
650
600
550
500
450
400
350
300
250
200
150
Мощность независимого множества
500 000
700 000
900 000
1 100 000
1 300 000
1 500 000
1 700 000
1 900 000
2 100 000
2 300 000
Номер итерации
|I|
|I|-i_del
мощность I упала до значения граница_спада_мощности_I,
поэтому i_del присваивается значение 1, а переменной грани-ца_спада_мощности
_I присваивается меньшее значение.
определенное время не было улучшения, значение i_del стало очень большое, а
поэтому поиск начинается с нового начального множества

РИС. 5

Заключение. Результаты проведенных вычислительных экспериментов показали, что с помощью предлагаемого алгоритма для каждого графа получены известные рекордные значения мощности независимого множества. Кроме того, для графа 1*zc*8192 улучшено рекордное значение мощности независимого множества.

І.П. Градинар

НАБЛИЖЕНИЙ АЛГОРИТМ РОЗВ'ЯЗАННЯ ЗАДАЧІ ЗНАХОДЖЕННЯ МАКСИМАЛЬНОЇ НЕЗАЛЕЖНОЇ МНОЖИНИ ВЕРШИН ГРАФА

Пропонується наближений алгоритм розв'язання задачі знаходження максимальної незалежної множини вершин графа. За допомогою цього алгоритму покращено відоме рекордне значення потужності максимальної незалежної множини для одного з графів.

I.P. Gradinar

AN APPROXIMATE ALGORITHM FOR SOLVING THE PROBLEM OF MAXIMUM INDEPENDENT SET IN A GRAPH

In the paper, an approximate algorithm for solving the problem of finding a maximum independent set in a graph is proposed. With the help of this algorithm the known record value of cardinality of the maximum independent set is improved for one of the graphs.

1. *Сергиенко И.В., Шило В.П.* Задачи дискретной оптимизации: проблемы, методы решения, исследования. – Киев: Наук. думка, 2003. – 264 с.
2. *Balaji S., Swaminathan V., Kannan K.* A Simple Algorithm to Optimize Maximum Independent Set // Advanced Modeling and Optimization. – 2010. – **12.** – P. 107–118. (http://camo.ici.ro/journal/vol12/v12a9.pdf)
3. *Градинар І. П.* Наближені алгоритми знаходження перешкодозахищених кодів, що коригують одну помилку в *Z*-каналі // Наук. вісн. Ужгород. ун-ту. Сер. математика і інформатика. – 2009. – Вип. 18. – С. 20–30.
4. *Сергиенко И.В., Шило В.П., Стецюк П.И.* Приближенный алгоритм решения задачи нахождения максимального независимого множества // Компьютерная математика. – Киев: Ин-т кибернетики им. В.М. Глушкова НАН Украины, 2000. – С. 4–20.
5. *Andrade D., Resende M.G.C., Werneck R.* Fast local search for the maximum independent set problem // Proceedings of 7th Workshop on Experimental Algorithms (WEA 2008), C.C. McGeoch (Ed.), LNCS. – 2008 – **5038**. – P. 220–234. (http://www2.research.att.com/~mgcr/doc/ls-indepset.pdf)

6. *Сергиенко И.В., Шило В.П.* Проблемы дискретной оптимизации: сложные задачи, основные подходы к их решению // Кибернетика и систем. анализ. – 2006. – № 4. – С. 3–25.
7. *Butenko S., Pardalos P.M., Sergienko I.V., Shylo V. and Stetsyuk P.* Finding maximum independent sets in graphs arising from coding theory // Symposium on Applied Computing, Madrid, Spain, ACM Press. – 2002. – P. 542–546.
8. *Sloane N. J. A.* Challenge problems: Independent sets in graphs, 2000. Last update Nov 18 2009.
http://www.research.att.com/~njas/doc/graphs.html

Получено 29.04.2010

Об авторе:

Градинар Иван Петрович,
аспирант Ужгородского национального университета.
e-mail: vgradinar@gmail.com