УДК 519.854

**В.П. ШИЛО, О.В. ШИЛО**

# РЕШЕНИЕ ЗАДАЧИ О МАКСИМАЛЬНОМ РАЗРЕЗЕ ГРАФА МЕТОДОМ ГЛОБАЛЬНОГО РАВНОВЕСНОГО ПОИСКА

**Ключевые слова:** *разрез графа, приближенные методы, метод глобального равновесного поиска, вычислительный эксперимент, эффективность алгоритма.*

Задача о максимальном разрезе графа (MAXCUT), вызывающая в последние два десятилетия большой интерес у исследователей, является классической проблемой дискретной оптимизации с многочисленными практическими приложениями [1–3]. Для ее решения в данной работе предлагается алгоритм, основанный на использовании метода глобального равновесного поиска (ГРП) [4], который нашел применения при решении различных классов задач дискретной оптимизации сложной природы [5–8]. Проведено сравнительное исследование эффективности предложенного алгоритма и лучших, известных на данный момент, приближенных алгоритмов решения рассматриваемой задачи, подтвердившее преимущества алгоритма ГРП как по быстродействию, так и по качеству получаемых решений.

Пусть задан неориентированный граф $G = G(V, E)$ с множеством вершин $V$, множеством ребер $E$ и каждому ребру $(i, j) \in E$ графа поставлено в соответствие число $w_{ij}$, называемое весом ребра $(i, j)$. Предположим, что $(V_1, V_2)$ — разбиение множества $V$ вершин графа $G$ на два непересекающихся подмножества — $V_1$ и $V_2$. Тогда подмножество ребер $(i, j) \in E$, обладающих свойством $i \in V_1$, $j \in V_2$, называется разрезом $(V_1, V_2)$ графа $G$. Очевидно, что любое такое разбиение порождает разрез графа.

Задача о максимальном разрезе неориентированного графа $G$ состоит в отыскании такого разбиения множества $V$ вершин на непересекающиеся подмножества $V_1$ и $V_2$, чтобы сумма весов $w(V_1, V_2) = \sum_{i \in V_1, j \in V_2, (i,j) \in E} w_{ij}$ ребер соответствующего разреза была максимальной.

Остановимся кратко на анализе современного состояния методов решения рассматриваемой задачи, которая является *NP*-трудной даже в случае, когда все ребра имеют единичный вес. Поэтому объем вычислений в точных алгоритмах обычно экспоненциально растет с ростом размерности задачи, что дает им возможность практически решать только задачи малых или средних размерностей.

Для задач больших размерностей эффективны приближенные методы, которым посвящено много публикаций, например [3, 9–11]. В работе [9] предложен алгоритм CirCut. Проведенные с этим алгоритмом экспериментальные расчеты показали, что по качеству получаемых решений и времени их нахождения он превосходит все известные на момент его появления приближенные алгоритмы.

Предложенные в [10] шесть алгоритмов базируются на методологии поиска решения с переменной окрестностью (VNS), жадной адаптивной поисковой процедуры (GRASP) и процедуры соединения полученных решений путями (Path Relinking). Показано, что с помощью алгоритма VNSPR [10] отыскиваются качественные решения за счет очень больших вычислительных затрат. В работе [12] рассмотрен рандомизированный алгоритм, гарантированная точность которого при

положительных весах составляет 0,878 от оптимального значения целевой функции. Однако этот алгоритм требует большого объема вычислений. Так, для решения задачи с числом вершин $n = 200$ ему необходимо около трех часов машинного времени. Две модификации, RRT и MST, мультистартного алгоритма табу описаны в [11]. В работе [13] предложен алгоритм SS, базирующийся на использовании метода разброса. Все эти алгоритмы на настоящий момент являются лучшими. Много важных результатов, касающихся решения задачи о максимальном разрезе графа, приведено в обзоре [3].

Введем следующие переменные:

$$x_i = \begin{cases} 0, \text{ если } i \in V_1, \\ 1, \text{ если } i \in V_2, \ i = 1, \ldots, n. \end{cases}$$

Тогда задачу о максимальном разрезе графа можно сформулировать как задачу булева квадратичного программирования без ограничений (UBQP): найти

$$\max_{x_i \in \{0,1\}} \left\{ f(x) = \sum_{(i,j) \in E} w_{ij} (x_i - x_j)^2 \right\}. \tag{1}$$

В то же время задача о максимальном разрезе графа сводится к взвешенной задаче о максимальной выполнимости WMAX-2-SAT. Действительно, поставим в соответствие каждому ребру $(i, j)$ две логические формулы: $x_i \vee x_j$ и $\bar{x}_i \vee \bar{x}_j$ с весом $w_{ij}$. Теперь можно показать, что рассматриваемый в задаче MAXCUT граф $G$ имеет разрез веса $w^*$ тогда и только тогда, когда соответствующая взвешенная задача о максимальной выполнимости имеет решение с суммарным весом $w^* + \sum_{(i,j) \in E} w_{ij}$.

В работах [7, 8] на основе общей схемы метода ГРП предложены алгоритмы решения задачи булева квадратичного программирования и взвешенной задачи о максимальной выполнимости (WMAX-SAT), которые показали высокую эффективность при решении этих задач. Полученные результаты позволяют надеяться на успех и при использовании метода ГРП для задачи о максимальном разрезе неориентированного графа.

Очевидно, что сведение рассматриваемой задачи к названным двум моделям и применение к ним известных алгоритмов ГРП [7, 8] не было бы столь эффективным, поэтому для задачи о максимальном разрезе неориентированного графа разработан специальный алгоритм, основанный на использовании метода ГРП и специфики этой задачи.

Наилучшее из всех допустимых решений задачи, рассмотренных к данному шагу вычислительного процесса, будем называть рекордным, а соответствующее ему значение целевой функции — рекордом.

Поскольку общая схема метода ГРП (GES) подробно описана в [5], представим базирующийся на ее использовании алгоритм решения рассматриваемой задачи в виде следующей процедуры:

**procedure** GES

1 *initialization_ algorithm's_ parameters*$(ngen, maxnfail, \mu, K)$,
$\mu$ — vector of temperature values, $K$ — number of temperature stages,
*maxnfail* — restart parameter, *ngen* — # of solutions generated during
each stage, $P = \varnothing$ — set of forbidden solutions, $x_{best} \leftarrow 0$ (zero vector)
2 **while** (stopping criterion = **FALSE**) **do**
3 **if** ($\widetilde{S} = \varnothing$) **then** {$\widetilde{S}$ — множество известных решений}
4 $x \leftarrow$ *construct_ random_ solution*
5 $x_{\max} \leftarrow x$;

```
6        S̃ ← x_max  (set of found solutions)
7     end if
8     nfail ← 0
9     while (nfail < maxnfail AND f(x_max) ≥ f(x_best) − Δ) do
10       x_oldmax ← x_max
11       for k = 0 to K do
12          calculate_generation_probabilities(pr^k, S̃)
13          for g = 0 to ngen do
14             x ← generate_solution(x_max, pr^k, max_dist_k)
15             R ← search_method(x)
16             R ← R \ P
17             S̃ ← S̃ ∪ R
18             x_good ← arg max_{x∈S̃} f(x)
19             if f(x_good) ≥ f(x_max) then
20                x_max ← x_good
21                if f(x_max) ≥ f(x_best) then x_best ← x_max
22             end if
23          end for
24       end for
25       if (f(old_x_max) = f(x_max)) then nfail ← nfail + 1
26       else nfail ← 0
27       S̃ = {x_max}
28    end while
29    P = P ∪ x_max
30    S̃ = ∅
31 end while
end GES
```

$$stopping\ \ criterion = \{processing\ \ time > 360\,sec.\ \ \mathbf{OR}\ \ f(x_{best}) \ge record\_f\ \}$$

Рассмотрим более детально эту процедуру. Внешний цикл (цикл итераций, строки 2–31) обеспечивает повторяемость поиска наилучшего решения. В нем либо проводится заданное количество повторов, либо он выполняется до установления истинности некоторого критерия останова (например, нахождения рекорда, не хуже известного).

Основным в структуре алгоритма ГРП является «температурный» цикл (строки 11–24), для которого необходимо задать значения величин $K$ и «температуры» $\mu_k$, $k = 0, \dots, K$. В этом цикле проводится серия стартов поиска наилучшего решения для возрастающих значений «температуры». «Температурный» цикл и его повторения позволяют гибко чередовать режимы сужения и расширения области поиска решения, что в итоге приводит к высокой эффективности метода ГРП.

Пусть $\widetilde{S}$ — подмножество множества $S$ допустимых решений рассматриваемой задачи, найденных алгоритмом ГРП,

$$\widetilde{S}_j^1 = \{x \mid x \in \widetilde{S},\ \ x_j = 1\},\ \ \widetilde{S}_j^0 = \{x \mid x \in \widetilde{S},\ \ x_j = 0\},\ \ j = 1, \dots, n.$$

Если множество $\widetilde{S}$ пусто, то операторы строк 4–6 позволяют найти первое решение и инициализировать необходимые данные. Следует отметить, что найденные алгоритмом ГРП решения не запоминаются, а используются для вычисления таких величин:

$$\widetilde{E}_{kj}^{l} = \frac{\sum\limits_{x \in \widetilde{S}_j^l} f(x) \exp\{-\mu_k f(x)\}}{\sum\limits_{x \in \widetilde{S}_j^l} \exp\{-\mu_k f(x)\}},\ \ k = 0, \dots, K,\ \ l = 0, 1,\ \ j = 1, \dots, n.$$

Эти величины позволяют определить компоненты вектора вероятности $\tilde{p}^k = (\tilde{p}_1^k, \dots, \tilde{p}_n^k)$ в соответствии со следующей формулой (см., например, [5]):

$$\tilde{p}_j^k = \frac{1}{1 + \frac{1-\tilde{p}_0^k}{\tilde{p}_0^k} \exp\left\{-0{,}5 \sum_{i=0}^{k-1} (\tilde{E}_{ij}^0 + \tilde{E}_{i+1\,j}^0 - \tilde{E}_{ij}^1 - \tilde{E}_{i+1\,j}^1)(\mu_{i+1} - \mu_i)\right\}}. \qquad (2)$$

Очевидно, что вероятности $\tilde{p}^k$ генерирования начальных решений (строка 12) для процедуры *генерация_решения* $(x)$ (строка 14) вычисляются исходя из понятий, заимствованных из метода отжига, и зависят от текущей «температуры» $\mu_k$ и найденного ранее множества $\tilde{S}$ допустимых решений.

Определяющие кривую отжига значения $\mu_k$, $k = 0, \dots, K$, обычно вычисляются по формулам $\mu_0 = 0$, $\mu_{k+1} = \alpha\mu_k$, $k = 1, \dots, K-1$. Величины $\mu_1$ и $\alpha > 1$ подбираются так, чтобы $||x_{\max} - \tilde{p}^K|| \cong 0$, где $x_{\max}$ — рекордное решение. Кривая отжига универсальна и не настраивается на отдельную задачу, а используется при решении всех задач. Масштабируются только коэффициенты целевой функции таким образом, чтобы значение рекорда равнялось заданной величине. Все компоненты вектора $\tilde{p}^0$ одинаковы.

Улучшающий цикл (строки 9–28) выполняется до тех пор, пока *maxnfail* раз не произойдет улучшения решения $x_{\max}$. Процедура *calculate_ generation_ probabilities* (строка 12) позволяет вычислять вероятности случайного возмущения решения $x_{\max}$ по формуле (2). Цикл нахождения новых решений (строки 13–23) повторяется заданное число *ngen* раз. С помощью процедуры *генерация_решения* случайным образом генерируется решение $x$, которое является начальным для процедуры *метод_поиска* (строка 16). При малых «температурах» $\mu_k$ возмущение решения $x_{\max}$ носит случайный, равновероятный характер, при больших «температурах» значения компонентов, одинаковых для лучших найденных решений, изменяются мало. «Температурный» цикл позволяет осуществлять диверсификацию поиска решения (при малых «температурах») и его интенсификацию (при больших «температурах»). Таким образом, по мере повышения «температуры» генерируемые решения приобретают признаки, присущие рекордным решениям, и в итоге сходятся к решению $x_{\max}$.

Приведем процедуру *генерация_решения*:

```
procedure генерация_ решения (x_max, pr^k, max_ dist_k)
1  x ← x_max ; dist = 0; j ← 1
2  while (j ≤ n )
3     if x_j = 1 then
4        if (p̃_j^k < random[0,1]) then
5              x_j ← 0; dist ← dist + 1
6              g(x) ← пересчет_вектора_ gains(x)
7        end if
8     end if
9     else
10       if (p̃_j^k ≥ random[0,1]) then
11             x_j ← 1; dist ← dist + 1
12             g(x) ← пересчет_вектора_ gains(x)
13       end if
14    end else
15    j ← j + 1
16    if (dist = max_ dist_k) then break
17 end while
```

В строке 1 этой процедуры происходит присвоение начальных значений вектору $x$ и переменным *dist* и $j$, в цикле строк 2–17 — возмущение вектора $x_{\max}$. Случайная, равномерно распределенная на отрезке [0,1] величина *random*[0,1] (строки 4 и 10) используется для моделирования случайных событий, ведущих к изменению решения $x_{\max}$. Если расстояние по Хэммингу между $x_{\max}$ и возмущенным решением $x$ становится равным заданному параметру $max\_\ dist_k$ (строка 16), цикл возмущения (строки 2–17) заканчивается. Параметр $max\_\ dist_k$ может задаваться динамически и зависеть от «температурного» индекса $k$.

При выборе алгоритма для процедуры *метод_поиска* учитывается специфика решаемой задачи. Используется алгоритм, позволяющий интенсифицировать поиск и эффективно исследовать окрестности начального решения в целях его улучшения. В процедуре *метод_поиска* использовались рандомизированные алгоритмы табу и локального поиска LocS. С помощью этой процедуры находится множество локальных (в окрестности определенного радиуса) решений, которые формируют множество *R*. При реализации рассматриваемого алгоритма поиск запрещенных решений блокируется, а пересчет осуществляется по мере нахождения новых решений $x \in R$. В строке 20 определяется решение $x_{\max}$, в строке 21 — лучшее решение $x_{best}$, найденное алгоритмом ГРП. После окончания улучшающего цикла (строки 9–28) к множеству *P* запрещенных точек добавляется точка $x_{\max}$ (строка 29).

Рассмотрим алгоритмы, используемые в процедуре *метод_поиска*.

Реализованы и исследованы две версии метода локального поиска. Они основаны на использовании оператора инверсии, который изменяет значение отдельной булевой переменной $x_i$ на ее отрицание $\neg x_i$, т.е. $x_i$ на $1-x_i$ для $i \in \{1,\dots,n\}$. Таким образом, в этих алгоритмах использовалась окрестность единичного (по Хэммингу) радиуса.

Следует отметить, что после каждого изменения вектора $x$ пересчитывается $n$-мерный вектор $g(x)$, определенный в окрестности единичного радиуса с центром в точке $x$: $g_j(x) = f(x_1,\dots,x_{j-1}, 1-x_j, x_{j+1},\dots,x_n) - f(x_1,\dots,x_j,\dots,x_n)$. Пересчет этого вектора соответствует идеологии метода вектора спада [5] и позволяет для многих задач экономить значительные вычислительные ресурсы. Так, для задачи булева квадратичного программирования без ограничений вида (1) вычисление целевой функции требует $O(n^2)$ арифметических операций, а пересчет вектора $g(x)$ — только $O(n)$ таких операций.

Самый простой алгоритм локального типа, используемый в проведенных исследованиях, — локальный поиск LocS. Представим его в виде следующей процедуры:

```
procedure LocS(x, g(x))
1  repeat
2        Generate random permutation RP of the set {1,…,n }
3        Δ = 0
4        for k = 1 to n do
5            j → RP[k]
6            if g_j(x) ≥ 0 then
7                Δ = Δ + g_j(x)
8                x_j = 1 − x_j
9                g(x) ← recalculate gains(x)
10           end if
11       end for
12 until Δ > 0 OR ∀ j, 1 ≤ j ≤ n: g_j(x) > 0
13 return x
```

Из строк 2, 5 данной процедуры вытекает, что компоненты вектора $x$ выбираются случайным образом, а из строки 6 следует, что переход осуществляется и в точки, не улучшающие значение целевой функции.

Вторым алгоритмом поиска, исследованным в работе, является алгоритм табу [8], который можно представить в виде следующей процедуры:

**procedure** Tabu($x, g(x), n_{bad}, tabu$)
1 $x_{\max} = x$; $step = 0$; $nfail = 0$;
2 **while** (($nfail < 9$ **AND** $f(x_{\max}) \geq f(x_{best})$) **OR** $nfail < 3$) **do**
3 $\Delta_{\max} = 0$; $\Delta_{cur} = 0$; $step_{impr} = step$
4 $M = \{1, \ldots, n\}$; $last\_used(j) = -\infty$, $j \in M$
5 **repeat**
6 **repeat**
7 $\Delta = 0$
8 Generate random permutation $RP$ of the set $M$
9 **for** $k = 1$ **to** $|M|$ **do**
10 $j = RP[k]$
11 **if** $(step - last\_used(j) > tabu$ **OR** $f(x) + g_j(x) > f(x_{\max}))$ **then**
12 **if** $g_j(x) \geq 0$ **then**
13 $\Delta = \Delta + g_j(x)$; $last\_used(j) = step$; $x_j = 1 - x_j$
14 $g(x) \leftarrow$ recalculate gains($x$); $step = step + 1$
15 **end if**
16 **end if**
17 **end for**
18 $\Delta_{cur} = \Delta_{cur} + \Delta$
19 **until** $\Delta > 0$
20 **if** $\Delta_{cur} \geq \Delta_{\max}$ **then**
21 $x_{\max} = x$
22 **if** $\Delta_{cur} > \Delta_{\max}$ **then**
23 $\Delta_{\max} = \Delta_{cur}$; $step_{impr} = step$; $nfail = 0$; **break;**
24 **end if**
25 **end if**
26 $\Delta = -\infty$
27 **for** $k = 1$ **to** $|M|$ **do**
28 $j = RP[k]$
29 **if** $(step - last\_used(j) > tabu$ **OR** $f(x) + g_j(x) > f(x_{\max}))$ **then**
30 **if** $g_j(x) \geq \Delta$ **then**
31 $\Delta = g_j(x)$; $ind_{best} = j$
32 **end if**
33 **end if**
34 **end for**
35 $\Delta_{cur} = \Delta_{cur} + \Delta$; $last\_used(ind_{best}) = step$; $x_{ind_{best}} = 1 - x_{ind_{best}}$
36 $g(x) \leftarrow$ recalculate gains($x$); $step = step + 1$
37 **if** $\Delta_{cur} \geq \Delta_{\max}$ **then**
38 $x_{\max} = x$
39 **end if**
40 **until** $step - step_{impr} < n_{bad}$
41 **if** $(step - step_{impr} \geq n_{bad})$ **then**
42 $x = x_{\max}$; $g(x) \leftarrow$ recalculate gains ($x$)
43 $nfail = nfail + 1$
44 **end if**
45 **end while**
46 **return** $x_{\max}$
**end** Tabu

Основной особенностью этого варианта алгоритма табу является цикл рестартов из наилучшего найденного решения (строки 2–45), что позволяет интенсифици-

ровать поиск. В строке 2 $x_{best}$ — лучшее найденное решение. Шаги алгоритма табу выполняются до тех пор, пока не будет никакого улучшения значения целевой функции на последних $n_{bad}$ итерациях (строки 5–40).

Перейдем теперь к описанию вычислительных экспериментов, проведенных в целях исследования эффективности различных приближенных алгоритмов. Алгоритм ГРП реализован на языке C++, все расчеты проводились с использованием PC с Intel® Core QUAD CPU Q9550 2.83GHz и 3.0GB оперативной памяти. Параметры алгоритма ГРП были одинаковыми во всех вычислительных экспериментах: $K = 21$, $\Delta = 4$, $max\_dist_k$: $max\_dist_0 = 0,5n$, $max\_dist_i = max\_dist_0 + i\left(\frac{10 - max\_dist_0}{K-1}\right)$, $i = 1,\ldots,K-2$, $max\_dist_{K-1} = 20$; $ngen = 81$ (для алгоритма LocS $ngen = 243$), $maxnfail = 1$. В начале каждого «температурного» цикла все компоненты вектора $\tilde{p}^0$ равны 0,5. Для «температурного» расписания использовались следующие значения: $\mu_0 = 0$, $\mu_1 = 3 \cdot 10^{-6}$, $\mu_k = \mu_{k-1} \frac{\log 10 - \log \mu_1}{48}$ для $k = 2,\ldots,20$, а для алгоритма табу — $n_{bad} = n/10$; $tabu = 21$.

Экспериментальные расчеты проводились для двух множеств тестовых задач: 24 задачи G1, G2, G3, G11, … , G16, G22, G23, G24, G32, … , G37, G43, G44, G45, G48, G49 и G50, приведенные в работе [14], а также 20 задач sg3dl101000, … ... , sg3dl1010000, sg3dl141000, … , sg3dl1410000 из [9]. Эти задачи использовались ранее авторами работ [9–13] для проверки эффективности их алгоритмов, что позволило максимально расширить множество сравниваемых алгоритмов.

Экспериментальные расчеты были выполнены в два этапа. На первом этапе сравнивались два варианта метода ГРП — GES LocS и GES Tabu. Результаты этого исследования представлены в табл. 1. В ней (и в табл. 2–4) приняты следующие обозначения: BKS — известный из литературы рекорд для задачи, BFS — рекорд, найденный соответствующим алгоритмом. В колонке BFS GES табл. 1 содержатся лучшие рекорды, полученные методом ГРП (двумя вариантами). Полужирным шрифтом выделены лучшие результаты. Если GES улучшил рекорд, известный в литературе, он выделен полужирным курсивом. Value в данной таблице обозначает среднее значение рекорда, найденное в результате десятикратного решения каждой задачи при различных начальных параметрах алгоритма. В двух строках табл. 1 со знаком $\sum$ для каждого множества тестовых задач приведены суммарные значения указанных в соответствующих колонках величин.

Анализ представленных в табл. 1 результатов показал, что каждый вариант метода ГРП имеет свои достоинства. С помощью первого варианта (GES LocS) успешно решены задачи первого множества — практически за одинаковое время со вторым вариантом (GES Tabu) он находил более качественные решения. Однако для тестовых задач второго множества вариант GES Tabu превзошел первый вариант как по быстродействию, так и по качеству найденных решений.

Проведенные на втором этапе экспериментальные исследования состоят из двух частей. Их результаты отображены в табл. 2–4. В первой части алгоритм ГРП сравнивался с алгоритмами SS [13], CirCut [9], VNSPR [10], а во второй — с алгоритмами RRT и MST [11].

При проведении вычислительных экспериментов с алгоритмами RRT и MST [11] каждая тестовая задача первого множества решалась десять раз, причем время решения было ограничено 1800 секундами для графов с числом вершин $n = 800$ и 3600 секундами — для графов с $n \geq 1000$. Каждая из задач второго множества решалась пять раз, при этом время решения ограничивалось 3600 секундами для задач малых размерностей и 7200 секундами для задач больших размерностей. Алгоритмы реализованы на языке C с использованием PC Pentium III 800.

**Таблица 1**

| Задача | Размерность | BKS | BFS | | | Время, с | | Value | |
|---|---|---|---|---|---|---|---|---|---|
| | | | GES | GES Tabu | GES LocS | GES Tabu | GES LocS | GES Tabu | GES LocS |
| G1 | 800 | **11624** | **11624** | **11624** | **11624** | **2.91** | 4.80 | **11624.0** | **11624.0** |
| G2 | 800 | **11620** | **11620** | **11620** | **11620** | **5.46** | 8.03 | **11620.0** | **11620.0** |
| G3 | 800 | **11622** | **11622** | **11622** | **11622** | **2.33** | 3.51 | **11622.0** | **11622.0** |
| G11 | 800 | **564** | **564** | **564** | **564** | **0.53** | 3.42 | **564.0** | **564.0** |
| G12 | 800 | **556** | **556** | **556** | **556** | **1.12** | 1.73 | **556.0** | **556.0** |
| G13 | 800 | **582** | **582** | **582** | **582** | **1.99** | 4.98 | **582.0** | **582.0** |
| G14 | 800 | 3063 | ***3064*** | ***3064*** | ***3064*** | **155.25** | 164.34 | 3063.3 | **3063.5** |
| G15 | 800 | **3050** | **3050** | **3050** | **3050** | 15.86 | **9.36** | **3050.0** | **3050.0** |
| G16 | 800 | **3052** | **3052** | **3052** | **3052** | 32.82 | **27.58** | **3052.0** | **3052.0** |
| G22 | 2000 | 13358 | ***13359*** | ***13359*** | ***13359*** | **86.60** | 98.61 | **13359.0** | 13358.5 |
| G23 | 2000 | **13354** | 13342 | 13342 | 13342 | 97.67 | **40.77** | 13341.6 | **13342.0** |
| G24 | 2000 | 13331 | ***13337*** | ***13337*** | ***13337*** | 212.57 | **174.36** | 13333.8 | **13335.6** |
| G32 | 2000 | 1402 | ***1410*** | ***1410*** | ***1410*** | **56.46** | 158.86 | **1410.0** | 1409.2 |
| G33 | 2000 | 1376 | ***1382*** | ***1382*** | ***1382*** | **88.67** | **104.54** | **1381.2** | 1379.6 |
| G34 | 2000 | 1372 | ***1384*** | ***1384*** | ***1384*** | **60.93** | 164.65 | **1384.0** | 1383.6 |
| G35 | 2000 | 7672 | ***7684*** | ***7684*** | ***7684*** | **152.25** | 216.61 | 7680.3 | **7681.1** |
| G36 | 2000 | 7670 | ***7676*** | ***7676*** | ***7677*** | 166.75 | **139.63** | 7670.8 | **7672.6** |
| G37 | 2000 | 7681 | ***7689*** | ***7689*** | ***7689*** | **159.22** | 180.38 | 7683.6 | **7684.6** |
| G43 | 1000 | **6660** | **6660** | **6660** | **6660** | **15.11** | 17.34 | **6660.0** | **6660** |
| G44 | 1000 | **6650** | **6650** | **6650** | **6650** | 10.42 | **3.61** | **6650.0** | **6650** |
| G45 | 1000 | **6654** | **6654** | **6654** | **6654** | 49.98 | **42.64** | **6654.0** | **6654** |
| G48 | 3000 | **6000** | **6000** | **6000** | **6000** | **0.04** | 0.38 | **6000.0** | **6000** |
| G49 | 3000 | **6000** | **6000** | **6000** | **6000** | **0.17** | 0.32 | **6000.0** | **6000** |
| G50 | 3000 | **5880** | **5880** | **5880** | **5880** | **0.08** | 13.53 | **5880.0** | **5880** |
| $\sum$ | | **150793** | **150841** | **150841** | **150842** | **1375.1** | 1583.98 | 150821.6 | **150824.3** |
| sg3dl101000 | 1000 | **896** | **896** | **896** | **896** | **6.56** | 110.40 | **896.0** | 895.6 |
| sg3dl102000 | 1000 | **900** | **900** | **900** | **900** | **1.11** | 2.87 | **900.0** | 900.0 |
| sg3dl103000 | 1000 | **892** | **892** | **892** | **892** | **5.87** | 37.73 | **892.0** | 892.0 |
| sg3dl104000 | 1000 | **898** | **898** | **898** | **898** | **2.59** | 6.78 | **898.0** | 898.0 |
| sg3dl105000 | 1000 | **886** | **886** | **886** | **886** | **20.72** | 54.50 | **886.0** | 886.0 |
| sg3dl106000 | 1000 | **888** | **888** | **888** | **888** | **3.42** | 8.58 | **888.0** | 888.0 |
| sg3dl107000 | 1000 | **900** | **900** | **900** | **900** | **19.53** | 145.07 | **900.0** | 899.8 |
| sg3dl108000 | 1000 | **882** | **882** | **882** | **882** | **27.99** | 101.77 | **882.0** | 881.8 |
| sg3dl109000 | 1000 | **902** | **902** | **902** | **902** | **9.17** | 18.06 | **902.0** | 902.0 |
| sg3dl1010000 | 1000 | **894** | **894** | **894** | **894** | **5.63** | 17.75 | **894.0** | 894.0 |
| sg3dl141000 | 2744 | **2446** | **2446** | **2446** | 2444 | 174.01 | **160.00** | 2443.8 | 2438.8 |
| sg3dl142000 | 2744 | **2458** | **2458** | **2458** | 2456 | **152.90** | 204.03 | **2457.8** | 2454.8 |
| sg3dl143000 | 2744 | **2442** | 2440 | 2440 | 2438 | 196.93 | **163.01** | 2439.2 | 2437.0 |
| sg3dl144000 | 2744 | **2450** | **2450** | **2450** | 2444 | 129.74 | **99.80** | 2447.6 | 2443.4 |
| sg3dl145000 | 2744 | **2446** | **2446** | **2446** | **2446** | 131.22 | **91.24** | 2445.8 | 2443.2 |
| sg3dl146000 | 2744 | **2450** | **2450** | **2450** | 2448 | **133.66** | 175.91 | **2450.0** | 2446.4 |
| sg3dl147000 | 2744 | **2444** | **2444** | **2444** | 2442 | **141.64** | 207.36 | 2442.2 | 2439.0 |
| sg3dl148000 | 2744 | 2446 | ***2448*** | ***2448*** | 2444 | 121.17 | **112.46** | 2445.4 | 2442.6 |
| sg3dl149000 | 2744 | 2424 | ***2426*** | ***2426*** | 2422 | **62.90** | 163.57 | **2424.4** | 2421.2 |
| sg3dl1410000 | 2744 | **2458** | **2458** | **2458** | 2456 | 169.34 | **161.38** | 2456.0 | 2453.0 |
| $\sum$ | | 33402 | **33404** | **33404** | 33378 | **1516.10** | 2042.27 | **33390.2** | 33356.6 |

**Таблица 2**

| Задача | Размерность | BKS | BFS | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | GES | SS | CirCut | VNSPR | RRT | MST |
| G1 | 800 | **11624** | **11624** | **11624** | **11624** | 11621 | **11624** | **11624** |
| G2 | 800 | **11620** | **11620** | **11620** | 11617 | 11615 | **11620** | **11620** |
| G3 | 800 | **11622** | **11622** | **11622** | **11622** | **11622** | **11622** | **11622** |
| G11 | 800 | **564** | **564** | 562 | 560 | **564** | **564** | 562 |
| G12 | 800 | **556** | **556** | 552 | 552 | **556** | **556** | 552 |
| G13 | 800 | **582** | **582** | 578 | 574 | 580 | 580 | 576 |
| G14 | 800 | 3063 | ***3064*** | 3060 | 3058 | 3055 | 3042 | 3063 |
| G15 | 800 | **3050** | **3050** | 3049 | 3049 | 3043 | 3024 | **3050** |
| G16 | 800 | **3052** | **3052** | 3045 | 3045 | 3043 | 3026 | **3052** |
| G22 | 2000 | 13358 | ***13359*** | 13346 | 13346 | 13295 | 13235 | 13358 |
| G23 | 2000 | **13354** | 13342 | 13317 | 13317 | 13290 | 13246 | 13329 |
| G24 | 2000 | 13331 | ***13337*** | 13303 | 13314 | 13276 | 13241 | 13327 |
| G32 | 2000 | 1402 | ***1410*** | 1398 | 1390 | 1396 | 1384 | 1392 |
| G33 | 2000 | 1376 | ***1382*** | 1362 | 1360 | 1376 | 1358 | 1368 |
| G34 | 2000 | 1376 | ***1384*** | 1364 | 1368 | 1372 | 1362 | 1368 |
| G35 | 2000 | 7672 | ***7685*** | 7668 | 7670 | 7635 | 7590 | 7672 |
| G36 | 2000 | 7670 | ***7677*** | 7660 | 7660 | 7632 | 7577 | 7669 |
| G37 | 2000 | 7681 | ***7689*** | 7664 | 7666 | 7643 | 7589 | 7675 |
| G43 | 1000 | **6660** | **6660** | 6656 | 6656 | 6659 | **6660** | **6660** |
| G44 | 1000 | **6650** | **6650** | 6648 | 6643 | 6642 | **6650** | **6650** |
| G45 | 1000 | **6654** | **6654** | 6642 | 6652 | 6646 | **6654** | **6650** |
| G48 | 3000 | **6000** | **6000** | **6000** | **6000** | **6000** | | |
| G49 | 3000 | **6000** | **6000** | **6000** | **6000** | **6000** | | |
| G50 | 3000 | **5880** | **5880** | **5880** | **5880** | **5880** | | |
| sg3dl101000 | 1000 | **896** | **896** | 882 | 880 | 892 | 892 | **896** |
| sg3dl102000 | 1000 | **900** | **900** | 894 | 892 | **900** | 898 | **900** |
| sg3dl103000 | 1000 | **892** | **892** | 884 | 882 | 884 | 886 | 888 |
| sg3dl104000 | 1000 | **898** | **898** | 892 | 894 | 896 | 896 | 896 |
| sg3dl105000 | 1000 | **886** | **886** | 880 | 882 | 882 | 884 | 884 |
| sg3dl106000 | 1000 | **888** | **888** | 870 | 886 | 880 | 884 | **888** |
| sg3dl107000 | 1000 | **900** | **900** | 890 | 894 | 896 | 898 | 898 |
| sg3dl108000 | 1000 | **882** | **882** | 880 | 874 | 880 | 880 | 880 |
| sg3dl109000 | 1000 | **902** | **902** | 888 | 890 | 898 | 900 | **902** |
| sg3dl1010000 | 2744 | **894** | **894** | 886 | 886 | 890 | 890 | 892 |
| sg3dl141000 | 2744 | **2446** | **2446** | 2428 | 2410 | 2416 | 2378 | 2438 |
| sg3dl142000 | 2744 | **2458** | **2458** | 2418 | 2416 | 2416 | 2394 | 2448 |
| sg3dl143000 | 2744 | **2442** | **2442** | 2410 | 2408 | 2406 | 2394 | 2434 |
| sg3dl144000 | 2744 | **2450** | **2450** | 2422 | 2414 | 2418 | 2390 | 2436 |
| sg3dl145000 | 2744 | **2446** | **2446** | 2416 | 2406 | 2416 | 2380 | 2432 |
| sg3dl146000 | 2744 | **2450** | **2450** | 2424 | 2412 | 2420 | 2394 | 2440 |
| sg3dl147000 | 2744 | **2444** | **2444** | 2404 | 2410 | 2404 | 2384 | 2434 |
| sg3dl148000 | 2744 | 2446 | ***2448*** | 2416 | 2418 | 2418 | 2386 | 2434 |
| sg3dl149000 | 2744 | 2424 | ***2426*** | 2412 | 2388 | 2384 | 2362 | 2416 |
| sg3dl1410000 | 2744 | **2458** | **2458** | 2430 | 2420 | 2422 | 2402 | 2450 |

Для сравнения с модифицированным алгоритмом разброса SS использовались программные реализации алгоритмов CirCut и VNSPR. Все алгоритмы имели настройки, рекомендованные авторами. С их помощью решалась каждая тестовая задача обоих множеств без ограничения на лимит времени. Использовался PC 3.2 GHz Intel Xenon processor с 2.0 GB оперативной памяти.

Сводные результаты второго этапа экспериментальных расчетов по решению указанных тестовых задач различными приближенными алгоритмами представлены в табл. 2.

**Таблица 3**

| Задача | BFS | Время, с | | | |
|---|---|---|---|---|---|
| | | SS | CirCut | VNSPR | GES |
| G1 | 11624 | 139 | 352 | 22732 | ***2.17*** |
| G2 | 11620 | 167.2 | 283 | 22719 | ***2.79*** |
| G3 | 11622 | 180.1 | 330 | 23890 | ***3.26*** |
| G11 | 564 | 171.8 | 74 | 10084 | ***2.39*** |
| G12 | 556 | 241.5 | 58 | 10852 | ***3.23*** |
| G13 | 580 | 227.5 | 62 | 10749 | ***1.9*** |
| G14 | 3060 | 186.5 | 128 | 16734 | ***10.16*** |
| G15 | 3049 | 142.8 | 155 | 17184 | ***8.93*** |
| G16 | 3045 | 161.9 | 142 | 16562 | ***3.77*** |
| G22 | 13346 | 1335.8 | 493 | 197654 | ***51.03*** |
| G23 | 13317 | 1021.7 | 457 | 193707 | ***5*** |
| G24 | 13314 | 1191 | 521 | 195749 | ***12.31*** |
| G32 | 1398 | 900.6 | 221 | 82345 | ***7.94*** |
| G33 | 1376 | 925.6 | 198 | 76282 | ***9.3*** |
| G34 | 1372 | 950.2 | 237 | 79406 | ***2.8*** |
| G35 | 7670 | 1257.5 | 440 | 167221 | ***25.4*** |
| G36 | 7660 | 1391.9 | 400 | 167203 | ***40.4*** |
| G37 | 7666 | 1386.8 | 382 | 170786 | ***18.25*** |
| G43 | 6659 | 405.8 | 213 | 35324 | ***6.8*** |
| G44 | 6648 | 355.9 | 192 | 34519 | ***4.7*** |
| G45 | 6652 | 354.3 | 210 | 34179 | ***22.41*** |
| G48 | 6000 | 20.1 | 119 | 64713 | ***0.24*** |
| G49 | 6000 | 35.1 | 134 | 64749 | ***0.27*** |
| G50 | 5880 | 26.8 | 231 | 147132 | ***2.1*** |
| sg3dl101000 | 892 | 406.1 | 106.0 | 20409.0 | ***3.03*** |
| sg3dl102000 | 900 | 302.4 | 116.0 | 20873.0 | ***7.65*** |
| sg3dl103000 | 884 | 410.4 | 112.0 | 20574.0 | ***0.75*** |
| sg3dl104000 | 896 | 485.9 | 103.0 | 19786.0 | ***1.66*** |
| sg3dl105000 | 882 | 400.9 | 106.0 | 19160.0 | ***1.82*** |
| sg3dl106000 | 886 | 461.8 | 119.0 | 17872.0 | ***5.83*** |
| sg3dl107000 | 896 | 386.2 | 115.0 | 21044.0 | ***2.28*** |
| sg3dl108000 | 880 | 466.9 | 104.0 | 19760.0 | ***11.44*** |
| sg3dl109000 | 898 | 493.6 | 121.0 | 20930.0 | ***4.35*** |
| sg3dl1010000 | 890 | 352.8 | 111.0 | 20028.0 | ***2.87*** |
| sg3dl141000 | 2428 | 1320.6 | 382.0 | 188390.0 | ***35.28*** |
| sg3dl142000 | 2418 | 1121.1 | 351.0 | 187502.0 | ***2.51*** |
| sg3dl143000 | 2410 | 1215.8 | 377.0 | 190028.0 | ***3.06*** |
| sg3dl144000 | 2422 | 1237.2 | 356.0 | 198809.0 | ***4.95*** |
| sg3dl145000 | 2416 | 1122.5 | 388.0 | 196725.0 | ***3.55*** |
| sg3dl146000 | 2424 | 1213.9 | 331.0 | 189366.0 | ***3.69*** |
| sg3dl147000 | 2410 | 1230.6 | 381.0 | 187902.0 | ***2.94*** |
| sg3dl148000 | 2418 | 1132.0 | 332.0 | 194838.0 | ***3.41*** |
| sg3dl149000 | 2412 | 1213.9 | 333.0 | 193627.0 | ***20.47*** |
| sg3dl1410000 | 2430 | 1125.8 | 391.0 | 196454.0 | ***5.86*** |
| $\sum$ | | 29277.8 | 10767.0 | 3986552.0 | ***374.95*** |

Анализ результатов табл. 2 показывает, что по качеству найденных для задач первого множества решений алгоритм ГРП превосходит остальные, выбранные для сравнения алгоритмы. Кроме того, для тестовых задач G13, G14, G22, G24, G32–G37 с помощью алгоритма ГРП получены новые рекорды.

**Таблица 4**

| Задача | BFS | Время, с | | | $f_{ср}$ | |
|---|---|---|---|---|---|---|
| | | GES to BFS | RRT | MST | RRT | MST |
| G1 | 11624 | ***2.91*** | 15 | 147 | 11624 | 11610 |
| G2 | 11620 | ***5.46*** | 180 | 195 | 11620 | 11607 |
| G3 | 11622 | ***2.33*** | 54 | 278 | 11622 | 11611 |
| G11 | 564 | ***0.53*** | 819 | 213 | 564 | 558 |
| G12 | 556 | ***1.12*** | 736 | 181 | 554 | 547 |
| G13 | 580 | ***0.49*** | 640 | 414 | 579 | 571 |
| G14 | 3063 | ***117.20*** | 785 | 787 | 3037 | 3059 |
| G15 | 3050 | ***15.86*** | 581 | 1109 | 3018 | 3047 |
| G16 | 3052 | ***32.82*** | 683 | 916 | 3022 | 3048 |
| G22 | 13358 | ***73.14*** | 2039 | 1519 | 13221 | 13306 |
| G23 | 13329 | ***22.84*** | 2162 | 1634 | 13224 | 13302 |
| G24 | 13327 | ***157.47*** | 1381 | 1824 | 13223 | 13308 |
| G32 | 1392 | ***1.21*** | 1498 | 1290 | 1380 | 1385 |
| G33 | 1368 | ***1.71*** | 1054 | 812 | 1355 | 1357 |
| G34 | 1368 | ***1.34*** | 1803 | 1324 | 1359 | 1359 |
| G35 | 7672 | ***27.95*** | 1200 | 2863 | 7582 | 7668 |
| G36 | 7669 | ***98.19*** | 2557 | 2578 | 7571 | 7661 |
| G37 | 7675 | ***36.54*** | 1974 | 2384 | 7582 | 7669 |
| G43 | 6660 | ***15.11*** | 845 | 733 | 6660 | 6648 |
| G44 | 6650 | ***10.42*** | 1484 | 478 | 6650 | 6639 |
| G45 | 6654 | ***49.98*** | 800 | 596 | 6653 | 6640 |
| sg3dl101000 | 896 | ***6.56*** | 961 | 1755 | 888.4 | 889.6 |
| sg3dl102000 | 900 | ***1.11*** | 1438 | 2208 | 896.8 | 896.8 |
| sg3dl103000 | 888 | ***1.84*** | 578 | 1616 | 884.4 | 883.2 |
| sg3dl104000 | 896 | ***0.57*** | 1677 | 913 | 894.4 | 892.4 |
| sg3dl105000 | 884 | ***1.96*** | 2163 | 1722 | 881.6 | 881.2 |
| sg3dl106000 | 888 | ***3.42*** | 1735 | 1808 | 882.4 | 883.6 |
| sg3dl107000 | 898 | ***1.17*** | 1619 | 2203 | 896 | 894.8 |
| sg3dl108000 | 880 | ***3.37*** | 511 | 712 | 877.2 | 878.4 |
| sg3dl109000 | 902 | ***9.17*** | 1472 | 1391 | 896.4 | 890.8 |
| sg3dl1010000 | 892 | ***2.14*** | 1311 | 297 | 888.4 | 888 |
| sg3dl141000 | 2438 | ***39.46*** | 4833 | 5265 | 2377.6 | 2425.2 |
| sg3dl142000 | 2448 | ***3.78*** | 3035 | 4645 | 2392 | 2436.4 |
| sg3dl143000 | 2434 | ***19.76*** | 4286 | 3878 | 2382 | 2422.4 |
| sg3dl144000 | 2436 | ***8.36*** | 3279 | 3665 | 2384.8 | 2430.4 |
| sg3dl145000 | 2432 | ***5.13*** | 3701 | 5871 | 2376.4 | 2424.4 |
| sg3dl146000 | 2440 | ***13.46*** | 3614 | 2760 | 2388.4 | 2429.6 |
| sg3dl147000 | 2434 | ***13.26*** | 2146 | 5405 | 2377.2 | 2420.4 |
| sg3dl148000 | 2434 | ***5.24*** | 2864 | 5726 | 2382 | 2424.8 |
| sg3dl149000 | 2416 | ***24.16*** | 4258 | 4784 | 2360.8 | 2406.4 |
| sg3dl1410000 | 2450 | ***29.88*** | 3869 | 5099 | 2392.8 | 2433.2 |
| $\sum$ | | ***868.4*** | 72640 | 83998 | | |

Решение тестовых задач второго множества связано со значительными трудностями, что объясняется спецификой этих задач. Графы для задач сконструированы на основе кубических решеток, моделирующих изинговские спины в слюде, а решения в колонке BKS получены алгоритмом H2 [9], который специально разработан для подобных структур и применим только к ним. Кроме того, такое хорошее качество решений обеспечивается очень большим временем счета (например, для решения каждой из последних десяти задач потребовалось около 33000 секунд времени работы ЭВМ SGI Origin 2000 [9]). Несмотря на это, методом ГРП для задач sg3dl148000, sg3dl149000 найдены новые рекорды.

Табл. 3 отображает результаты первой части экспериментальных расчетов — скорость получения лучших решений алгоритмами SS, CirCut, VNSPR и GES. Во второй колонке содержатся наилучшие рекорды, найденные указанными алгоритмами. В следующих трех колонках приведено время, понадобившееся соответствующим алгоритмам для получения их рекордных решений (оно не пересчитывалось и приводится для соответствующих машин). С помощью алгоритма ГРП каждая задача решалась десять раз. В колонке GES приведено среднее время, затраченное на получение решений с рекордом, не хуже рекорда из колонки BFS, т.е. решений, которые по значению целевой функции не уступают решениям, совместно полученным остальными алгоритмами. Таким образом, анализ результатов табл. 3 подтверждает преимущества алгоритма ГРП над алгоритмами SS, CirCut, VNSPR и по скорости получения хороших решений.

Перейдем к рассмотрению второй части вычислительных экспериментов. В соответствии с планом экспериментов, описанных в [11], с помощью алгоритма ГРП десять раз решалась каждая задача из обоих множеств. Также решались по десять раз эти задачи с использованием алгоритмов RRT и MST. В табл. 4 приведены полученные данные. В колонках 3–5 для каждой задачи указано среднее время ее решения соответствующим алгоритмом, а в колонках 6, 7 — среднее значение рекорда ($f_{\text{ср}}$ ). Как видно, и в этом случае алгоритм ГРП превосходит алгоритмы RRT и MST как по качеству получаемых решений, так и по времени их нахождения.

Анализ приведенных результатов вычислительных экспериментов свидетельствует о высокой эффективности и конкурентоспособности метода ГРП при решении задачи о максимальном разрезе неориентированного графа. Более того, опыт, накопленный при решении других задач [5–8], позволяет надеяться, что для задач большей размерности преимущества алгоритмов ГРП будут только возрастать.

СПИСОК ЛИТЕРАТУРЫ


1. Barahona F., Grotschel M., Junger M., Reinelt G. An application of combinatorial optimization to statistical physics and circuit layout design // Oper. Res. — 1988. — **36**. — P. 493–513.
2. Chang K.C., Du D.H.-C. Efficient algorithms for layer assignment problem // IEEE Trans. Computer — Aided Design of Integrated Circuits and Systems. — 1987. — **6**. — P. 67–78.
3. Poljak S., Tuza Z. Maximum cuts and large bipartite subgraphs // DIMACS Series in Discrete Mathematics and Theoretical Computer Science. — 1995. — **20**. — P. 181–244.
4. Шило В.П. Метод глобального равновесного поиска // Кибернетика и системный анализ. — 1999. — № 1. — С. 74–81.
5. Сергиенко И.В., Шило В.П. Задачи дискретной оптимизации: проблемы, методы решения, исследования. — Киев: Наук. думка, 2003. — 264 с.
6. Сергиенко И.В., Шило В.П. Проблемы дискретной оптимизации: сложные задачи, основные подходы к их решению // Кибернетика и системный анализ. — 2006. — № 4. — С. 3–25.
7. Pardalos P., Prokopyev O., Shylo O., Shylo V. Global equilibrium search applied to the unconstrained binary quadratic optimization problem // Optim. Methods and Software. — 2008. — **23**. — P. 129–140.
8. Prokopyev O., Shylo O., Shylo V. Solving weighted MAX-SAT via global equilibrium search // Oper. Res. Lett. — 2008. — **36**, N 4. — P. 434-438.
9. Burer S., Monteiro R.D.C., Zhang Y. Rank-two relaxation heuristics for MAX-CUT and other binary quadratic programs // SIAM J. Optim. — 2002. — **12**. — P. 503–521.
10. Festa P., Pardalos P.M., Resende M.G.C., Ribeiro C.C. Randomized heuristics for the maxcut problem // Optim. Methods and Software. — 2002. — **17**. — P. 1033–1058.
11. Palubeckis G., Krivickiene V. Application of multistart tabu search to the Max-Cut problem // Information Technology and Control. — Kaunas: Technologija, 2004. — **2(31)**. — P. 29–35.
12. Goemans M.X., Williamson D.P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming // J. ACM. — 1995. — **42**. — P. 1115–1145.
13. Marti R., Duarte A., Laguna M. Advanced scatter search for the Max-Cut problem // INFORMS J. Computing. — 2009. — **21**, N 1. — P. 26–38.
14. Helmberg C., Rendl F. A spectral bundle method for semidefinite programming // SIAM J. Optim. — 2000. — **10**. — P. 673–696.


*Поступила 25.04.2010*