СИСТЕМНЫЙ АНАЛИЗ

В.П. ШИЛО, О.В. ШИЛО

УДК 519.854

РЕШЕНИЕ ЗАДАЧИ БУЛЕВА КВАДРАТИЧНОГО ПРОГРАММИРОВАНИЯ БЕЗ ОГРАНИЧЕНИЙ МЕТОДОМ ГЛОБАЛЬНОГО РАВНОВЕСНОГО ПОИСКА

Ключевые слова: *булево квадратичное программирование, приближенные методы, метод глобального равновесного поиска, вычислительный эксперимент, сравнительное исследование алгоритмов.*

ВВЕДЕНИЕ

Объектом исследования настоящей статьи является известный класс целочисленных оптимизационных задач вида

$$\max\{f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i x_j \mid x \in B^n\}, \tag{1}$$

где q_{ij} — элементы симметричной действительной матрицы Q порядка n, B^n — множество n-мерных векторов с компонентами 0 или 1. Такая задача называется задачей булева квадратичного программирования без ограничений (UBQP). Поскольку $x_i^2 = x_i$ для всех переменных $x_i \in \{0,1\}$, $i = 1, \ldots, n$, линейная функция cx (при ее наличии) может быть перенесена в квадратичную часть целевой функции в (1). К (1) сводится также квадратичная задача вида $\max\{xQx \mid Dx = b, x \in B^n\}$, где Q — матрица из (1), $D \in R^{m\times n}$, $b \in R^n$, R^n — множество n-мерных действительных векторов.

Многие фундаментальные задачи из области науки, инженерии, финансов, медицины и др. могут быть сформулированы как задачи булева квадратичного программирования. Квадратичные функции с булевыми переменными естественно возникают при моделировании отборов и взаимодействий. Рассмотрим множество из n объектов, каждый из которых может быть отобран или не отобран. Каждой паре (i, j) объектов соответствует вес q_{ij}, который измеряет взаимодействие между точками i и j. Считаем $x_i = 1$, если объект отобран, и $x_i = 0$ в противном случае. Сумма всех взаимодействий между отобранными точками, так называемое глобальное взаимодействие, может быть представлена как квадратичная булева функция $\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i x_j$. Класс таких задач исследовался в физике твердого тела. Кроме того, приложения версий задачи (1) с ограничениями и без ограничений могут быть найдены в многочисленных отраслях, включая медицину, ма-

шинное проектирование, планирование, управление сообщениями, химию [1, 2]. Многие задачи по теории графов могут быть представлены в терминах квадратичного булева программирования, включая хорошо изученные задачи нахождения максимальной клики и максимального разреза [3]. Кроме того, задача UBQP является общей моделью для широкого круга дискретных задач оптимизации [4]. В работе [5] даны примеры ее использования при рассмотрении задач раскраски вершин графа, упаковки, разбиения, линейного упорядочивания и т.п.

Задача UBQP принадлежит к классу *NP*-трудных задач дискретной оптимизации. Существует ограниченное количество ее подклассов, о которых известно, что они полиномиально разрешимы (например, [6]). Также известно, что задача «имеет ли задача UBQP единственное решение?» — *NP*-трудная [7]. Кроме того, задача булева квадратичного программирования остается *NP*-трудной, даже если известно, что глобальный оптимум единственный [7].

Поскольку задача (1) — *NP*-трудная, точные алгоритмы (например, [8, 9]) могут быть применимы для нее только при размерности несколько сотен переменных и умеренно разреженной матрице Q. Для задач больших размерностей с плотными матрицами пригодны только приближенные методы, позволяющие находить почти оптимальные решения за приемлемое время. Среди них следует отметить алгоритмы, основанные на методах табу [4, 10–12], отжига [11, 13, 14]. Кроме того, хорошо зарекомендовали себя такие алгоритмы, использующие популяции: эволюционные [15–18], меметические [19], рассеивания (scatter) [20].

Для задачи UBQP в работе [21] предложен алгоритм глобального равновесного поиска (ГРП), проведено его сравнение с лучшим на тот момент алгоритмом MST2 [12]. Хотя это сравнение и показало преимущество алгоритма ГРП (GES), но ввиду программной ошибки оно было проведено не на общепринятых тестах (в них были ошибочно обнулены диагональные элементы матрицы). За время, прошедшее с момента опубликования статьи [21], появились новые интересные алгоритмы (например [4]), изменился наш подход к построению и сравнению алгоритмов.

В настоящей статье на базе метода ГРП разработан новый алгоритм решения задачи UBQP, проведено его исследование и сравнение с известными алгоритмами с использованием общепринятых тестов (при увеличении их размерности до $n = 10000$).

АЛГОРИТМ ГРП

Целевая функция задачи (1) для различных тестов порождает сходный ландшафт. Он характеризуется наличием отдельных пиков приблизительно одной высоты, причем расстояние между ними довольно значительное. Если называть область притяжения этих пиков горной системой, то все они принадлежат разным горным системам, что исключает возможность перехода из одного пика в другой средствами локального поиска. Это свидетельствует о чрезвычайной сложности задач UBQP, особенно если в качестве критерия оценки качества оптимизации выбрать обязательное нахождение наилучшего известного решения (желательно за возможно меньшее время). Поэтому для создания хорошего алгоритма решения этих задач следует учитывать их особенности. Рассмотрим модификацию основной схемы метода ГРП [22, 23] для задачи (1).

Общая схема метода ГРП включает два этапа: генерацию решения и поиск локального максимума в окрестности этого решения.

Предположим, что множество S является подмножеством множества допустимых решений задачи (1), найденных методом ГРП, и

$$S_j^1 = \{ x \in S \mid x_j = 1 \},\ S_j^0 = \{ x \in S \mid x_j = 0 \},\ j = 1, \ldots, n.$$

Основным в структуре метода ГРП является «температурный» цикл. В нем проводится серия стартов поиска наилучшего решения для возрастающих значений температуры. Данный цикл и его повторения позволяют гибко чередовать режимы диверсификации и интенсификации области поиска решения, что в итоге приводит к высокой эффективности метода ГРП. Для температурного цикла необходимо задать K — число его выполнений и $\mu_0 < \mu_1 < \ldots < \mu_K$ — значения температуры с индексами, которые соответствуют номерам температурного цикла. Для $k = 0, \ldots, K$, $j = 1, \ldots, n$, $u \in \{0,1\}$ дополнительно определим такие величины:

$$E_{kj}^u = \begin{cases} 0, \text{ если } S_j^u = \varnothing, \\ \dfrac{\sum\limits_{x \in S_j^u} f(x) \exp\{\mu_k (f(x) - f(x_{max}))\}}{\sum\limits_{x \in S_j^u} \exp\{\mu_k (f(x) - f(x_{max}))\}}, \text{ если } S_j^u \neq \varnothing. \end{cases} \tag{2}$$

В методе ГРП множество S используется при случайной генерации начальных решений для применяемого поискового метода. Вектор температурных параметров $\mu = (\mu_0, \ldots, \mu_K)$, $\mu_0 < \mu_1 < \ldots < \mu_K$, позволяет управлять близостью расстояния генерируемых начальных решений к лучшему решению x_{max} в множестве S. Это происходит вследствие того, что генерируемое начальное решение получается случайным изменением компонент вектора x_{max} по следующему правилу: если j-я компонента вектора x_{max} равна нулю, она изменяется с вероятностью $p_j(\mu_k)$, в противном случае — с вероятностью $1 - p_j(\mu_k)$. Вероятности предлагаемой версии алгоритма рассчитывались по формулам

$$p_j(\mu_k) = \frac{1}{1 + \dfrac{1 - p_j(\mu_0)}{p_j(\mu_0)} \exp\left\{\dfrac{1}{2} \sum\limits_{i=0}^{k-1} (\mu_{i+1} - \mu_i)(E_{ij}^0 + E_{i+1\,j}^0 - E_{ij}^1 - E_{i+1\,j}^1)\right\}},$$

$$j = 1, \ldots, n, \ k = 1, \ldots, K. \tag{3}$$

Из (2) следует, что величины E_{kj}^u равны взвешенной сумме значений целевой функции по множеству известных решений, j-е компоненты которых равны u. Вес каждого решения зависит от значения его целевой функции и величины температурного параметра μ_k. Найденное новое решение не запоминается, а используется для пересчета величин E_{kj}^u.

Пусть $g_j^u = \max\{f(x) \mid x \in S_j^u\}$, $j = 1, \ldots, n$, $u \in \{0,1\}$. Из (2), (3) следует, что $\lim\limits_{\mu_k \to \infty} p_j(\mu_k) \to 1$ при $g_j^0 < g_j^1$, в противном случае $\lim\limits_{\mu_k \to \infty} p_j(\mu_k) \to 0$. Соотношения (2), (3) получены из аппроксимации распределения Больцмана [21–23]. Компоненты μ_k температурного вектора μ вычисляются по формулам $\mu_0 = 0$, $\mu_{k+1} = \alpha \mu_k$, $k = 1, \ldots, K-1$.

Рассмотрим схему алгоритма ГРП решения задачи UBQP.

Входные данные: μ — вектор температурных параметров, K — число выполнений температурного цикла, *maxnfail* — параметр рестарта, *ngen* — количество решений, генерируемых в каждом цикле.

Procedure GES:

1. $EliteSet = \varnothing$
2. while (stopping criterion = FALSE) do
3. $x \leftarrow$ construct random solution
4. $x^{max} = x;\ \ x^{best} = x$
5. $S = \{x^{max}\};\ nfail = 0;\ nrep = 0$
6. while $(nfail < maxnfail$ AND $f(x^{best}) = f(x^{max}))$ do
7. $x^{old} = x^{max};\ nrep = nrep + 1$
8. for $k = 0$ to $K - 1$ do
9. calculate generation probabilities $(p(\mu_k), S, \mu_k)$
10. for $g = 0$ to $ngen$ do
11. $x \leftarrow$ generate initial solution $(x^{max}, p(\mu_k), EliteSet)$
12. $R \leftarrow$ Tabu search method $(x, gains(x), EliteSet)$
13. $S = S \cup R$
14. $x^{max} = \arg\max\{f(x) \mid x \in S\}$
15. if $f(x^{max}) > f(x^{best})$ then $x^{best} = x^{max}$
16. end for
17. end for
18. $S = \{x^{max}\}$
19. if $f(x^{max}) \le f(x^{old})$ then $nfail = nfail + 1$
20. else $nfail = 0$
21. end while
22. $EliteSet = EliteSet \cup x^{max}$
23. end while
24. return x^{best}

Основной цикл схемы (строки 2–23) повторяется до выполнения некоторого критерия останова. При проведении экспериментов алгоритм прекращал работу, когда длительность его работы превышала некоторое предельное значение. Как отмечалось выше, основной элемент в структуре метода ГРП — температурный цикл (строки 8–17). Число K его выполнений и вектор температурных параметров $\mu = (\mu_0, \ldots, \mu_K)$ определены заранее. Выбор $\alpha > 0$ и μ_1 должен быть сделан таким образом, чтобы вектор вероятности $p(\mu_K)$ был близок к лучшему решению из множества S.

Вероятности, с которыми генерируются новые решения, вычисляются с использованием (2), (3) в начале каждого температурного цикла (строка 9). Для каждого вектора вероятности генерируется *ngen* решений (строка 11). Они используются в качестве начальных решений для процедуры Tabu search method (строка 12). Множество R решений, найденных с помощью этой процедуры, используется для пополнения множества S (строка 13). Как сказано выше, множество S используется в псевдокоде для упрощения описания алгоритма. При реализации алгоритма оно не запоминается, хранятся только значения E^u_{kj}.

Единственные решения, которые запоминаются алгоритмом, — так называемые элитные решения, составляющие множество *EliteSet* (строка 22). Предполагается, что вследствие цикла интенсификации поиска (строки 6–21) их окрестности хорошо исследованы и в них нет улучшающих решений. Таким образом, дальнейший поиск проводится только среди решений, для которых расстояние

Хемминга от множества *EliteSet* составляет не менее d_p. Поэтому алгоритм не ведет поиск в уже исследованных областях.

В алгоритме табу, применяемом при реализации второго этапа алгоритма ГРП, используется окрестность радиуса 1. Решение y принадлежит окрестности $N_1(x)$ радиуса 1 с центром в точке x, если расстояние $d(x, y)$ Хемминга равно единице. Другими словами, если $y \in N_1(x)$, то $y_j = 1 - x_j$ для некоторого индекса j и $y_k = x_k$ для $k \neq j$. Будем считать, что такое решение y найдено с использованием хода $m_j : y = m_j(x)$. Пусть $gains(x)$ — вектор, в котором j-я компонента представляет увеличение целевой функции, полученной с помощью хода m_j. Он может быть вычислен за линейное время

$$gains_j(x) = q_{jj}(\bar{x}_j - x_j) + 2\sum_{i=1, i\neq j}^{n} q_{ij}x_i(\bar{x}_j - x_j), \text{ где } \bar{x}_j = 1 - x_j.$$

Рассмотрим теперь схему алгоритма табу, который используется в алгоритме ГРП.

Входные данные: x — начальное решение, $g(x)$ — вектор *gains*, n_{bad} — количество итераций без улучшения решения, *tenure* — параметр, x^{max} — лучшее решение в множестве S, x^{best} — рекорд, найденный алгоритмом, *nrep* — число повторов цикла интенсификации поиска (строки 6—21) в Procedure GES.

Procedure Tabu search method:

1. $x^{good} = x;\ nfail = 0;\ step = 0;\ R = \varnothing;\ nbad = n/2;\ improve = 1$
2. if $nrep > 0$ then $mxnfail = 9$
3. else $mxnfail = 3$
4. repeat
5. $\quad step_{impr} = step;\ c = 0$
6. $\quad M = \{1,2,\ldots,n\};\ last\_used(j) = -\infty;\ tabu(j) = tenure + RANDOM(10);\ j = 1,\ldots,n$
7. $\quad$ repeat
8. $\quad\quad$ Generate a random permutation RP of M
9. $\quad\quad \Delta = -\infty$
10. $\quad\quad$ for $k = 1$ to n do
11. $\quad\quad\quad j = RP[k]$
12. $\quad\quad\quad$ if $(step - las\_used(j) > tabu(j) OR (f(x) + g_j(x) > f(x^{max})))$ then
13. $\quad\quad\quad\quad$ if $(g_j(x) > \Delta) AND (dist(move_j(x), EliteSet) \geq d_p)$ then
14. $\quad\quad\quad\quad\quad \Delta = g_j(x);\ ind = j$
13. $\quad\quad\quad\quad$ end if
14. $\quad\quad\quad$ end if
15. $\quad\quad$ end for
16. $\quad\quad$ if $(\Delta < 0$ AND $c \geq 0)$ then
17. $\quad\quad\quad x^{good} = x;\ c = 0$
18. $\quad\quad\quad$ if $f(x) > f(x^{max})$ then
19. $\quad\quad\quad\quad nbad = 5n;\ R = R \cup x;\ nfail = 0$
20. $\quad\quad\quad\quad$ if $f(x) > f(x^{best})$ then
21. $\quad\quad\quad\quad\quad mxnfail = 9$
22. $\quad\quad\quad\quad$ end if
23. $\quad\quad\quad\quad improve = 1;$ break;
24. $\quad\quad\quad$ end if

25. end if
26. $last\ used(ind) = step;\ x_{ind} = 1 - x_{ind};\ tabu(ind) = tenure + RANDOM(10)$
27. $g(x) \leftarrow$ recalculate $gains(x); c = c + \Delta; step = step + 1$
28. until $step - step_{impr} < nbad$
29. if $step - step_{impr} \geq nbad$ then
30. $x = x^{good}; g(x) \leftarrow$ recalculate $gains(x)$
31. $nfail = nfail + 1$
32. end if
33. end while
34. until $(improve = 0)$ OR $(nfail \geq mxnfail)$
35. return x^{good}, R

Рассмотрим процедуру генерации решений согласно вероятности, определенной формулой (3).

Входные данные: x — решение, к которому применяется случайное возмущение, $p(\mu_k)$ — вектор вероятности, рассчитанный по формулам (2), (3).

Function:

1. $dist = 0; j = 1$
2. while $(j \leq n)$ do
3. if $x_j = 1$ then
4. if $p(\mu_k) \leq random[0,1]$ then
5. $x_j = 0; dist = dist + 1$
6. end if
7. else
8. if $p(\mu_k) \geq random[0,1]$ then
9. $x_j = 1; dist = dist + 1$
10. end if
11. end if
12. $j = j + 1$
13. if $dist = dist_{max}$ then return x
14. end while
15. return x

На основании вектора x с помощью процедуры генерации выполняются операторы инверсии (строки 2–16). Равномерно распределенная в (0,1) переменная *random*[0,1] определяет возможность применения операторов инверсии к ней (строки 4,8). Процедура заканчивается, когда расстояние Хемминга между x и возмущенным решением равно параметру $dist_{max}$.

СРАВНИТЕЛЬНОЕ ИССЛЕДОВАНИЕ АЛГОРИТМОВ

Рассмотрим описание вычислительных экспериментов по решению алгоритмом ГРП тестовых задач вида (1) большой размерности, сравнение полученных результатов с результатами лучших известных алгоритмов.

1. Тестовые задачи. При проведении экспериментальных расчетов были выбраны 24 случайно сгенерированные задачи p3000.1,...,p10000.3 большой размерности ($3000 \leq n \leq 10000$) с плотностью заполнения матрицы от 0.5 до 1 [12]. Обычный набор тестов был расширен задачами размерности $n = 10000$, поскольку опыт, приобретенный за годы, прошедшие с момента опубликования статьи [21], позволил поднять планку эффективности для алгоритмов. Коды для генерирования этих задач доступны на сайте http: //www.soften.ktu.lt/~gintaras/ubqop_its.html. Именно эти задачи в настоящее время активно используются для проверки эффективности разработанных алгоритмов.

Таблица 1

Параметр	Место нахождения параметра	Смысловое значение	Величина параметра
K	Procedure GES (строка 8)	Число выполнений температурного цикла	6
ngen	Procedure GES (строка 10)	Количество генерируемых начальных решений	$\begin{cases}45, nrep = 0\\80, nrep > 0\end{cases}$
maxnfail	Procedure GES (строка 6)	Переменная цикла интенсификации поиска (строки 6–21)	1
tenure	Procedure Tabu search method (строки 6, 26)	Начальное значение переменной запрета в алгоритме табу	*n/150*
d_p	Procedure Tabu search method (строка 13), Procedure generate initial solution (строки 4, 8)	Минимальное разрешенное расстояние до множества элитных решений *EliteSet*	*200*
$dist_{max}$	Procedure generate initial solution (строка 12)	Максимальное количество случайных возмущений	$\begin{cases}n, nrep = 0\\d_p, nrep > 0\end{cases}$

Заметим, что тестовые задачи из библиотеки ORLIB при $50 \le n \le 2500$ и подобные задачи из сайта http://people.brunel.ac.uk/~mastjjb/jeb/orlib/bqpinfo.html здесь не рассматриваются, так как с помощью алгоритма ГРП они решаются достаточно быстро, не используя при этом все его возможности.

2. План вычислительных экспериментов. Алгоритм ГРП реализован на языке C++, все вычислительные эксперименты проводились с использованием PC Intel® Core QUAD CPU Q9550 2.83GHz и 3.0GB оперативной памяти. Значения основных параметров алгоритма ГРП приведены в табл. 1.

Начальные вероятности $p_j(\mu_0) = \frac{1}{2}, j = 1, \ldots, n,\ \mu_0 = 0$. Для температурного расписания использованы следующие значения: $\mu_1 = 10^{-7}$, $\mu_k = \mu_{k-1} \frac{\log 0.003 / coef - \log \mu_1}{4}$ при $k = 2, \ldots, K$, $coef = 1.8 * 10^8 / f(x^{BKS})$, где $f(x^{BKS})$ — известный для данной задачи рекорд.

Следует отметить, что приведенные ниже результаты вычислительных экспериментов получены без специальной настройки параметров, т.е. все используемые параметры постоянны для всех тестовых задач. Значения параметров *tenure* и *nbad* алгоритма табу из [4] подтверждают, что именно схема метода ГРП, а не применяемые методы поиска делает алгоритм ГРП столь эффективным.

Для сравнительного экспериментального исследования с алгоритмом ГРП выбран алгоритм MST2 [12], поскольку прежде всего реализация MST2 алгоритма табу на данное время является одной из лучших для решения задачи UBQP. Она показывает хорошую и устойчивую работу алгоритма на разных множествах эталонных тестов. Это следует из сравнительного исследования [12], где представлены результаты с использованием других известных подходов, включая методы табу, отжига, генетический локальный поиск. Кроме того, исходный текст алгоритма доступен в сети http://www.soften.ktu.lt/~gintaras/ubqop_its.html. Это позволило проводить вычислительные эксперименты, в которых для запуска алгоритмов использовался один и тот же персональный компьютер. Следовательно, представленное ниже время решения задач может служить одним из параметров для сравнения алгоритмов.

Каждая задача из представленного множества тестовых задач решалась обоими алгоритмами 20 раз при различных начальных значениях датчика случайных чисел. Алгоритм MST2 выполнял 1000 итераций; затраченное им на решение задачи время (см. табл. 2) служило критерием останова для алгоритма GES. Такой план экспериментов также позволяет провести сравнение с разработанным в последнее время перспективным алгоритмом HMA [4].

Таблица 2

Задача	$f(x^{BKS})$	t_{max}, сек	success		g_{avr}		t_{avr}		t_{best}	
			MST2	GES	MST2	GES	MST2	GES	MST2	GES
p3000.1	3931583	580	20	20	0.0	0.0	19.5	7.60	4.17	2.83
p3000.2	5193073	890	20	20	0.0	0.0	21.4	6.98	7.69	0.36
p3000.3	5111533	890	20	20	0.0	0.0	130.7	9.70	7.69	0.44
p3000.4	5761822	1120	20	19	0.0	19.25	40.4	9.12	12.92	0.47
p3000.5	5675625	1120	18	20	2.7	0.0	416.6	75.75	18.53	7.03
p4000.1	6181830	930	20	20	0.0	0.0	19.8	9.59	7.83	1.11
p4000.2	7801355	1400	20	20	0.0	0.0	262.1	86.40	16.44	6.55
p4000.3	7741685	1400	20	20	0.0	0.0	95.7	36.12	19.23	1.39
p4000.4	8711822	1720	20	20	0.0	0.0	119.5	16.67	24.13	0.97
p4000.5	8908979	1720	20	20	0.0	0.0	430.1	72.90	48.25	17.11
p5000.1	8559680	1320	0	6	396.9	234.75	408.37	391.70	–	325.27
p5000.2	10836019	1960	1	19	552.9	29.1	234.5	519.53	1228.11	25.14
p5000.3	10489137	1960	17	20	37.8	0.0	916.0	413.16	42.7	21.49
p5000.4	12252318	2360	1	8	700.3	291	1099.1	927.61	1778.45	219.45
p5000.5	12731803	2360	13	20	307.1	0.0	885.5	227.62	46.75	11.19
p6000.1	11384976	1740	20	19	0.0	12.15	232.8	125.21	68.23	28.11
p6000.2	14333855	2550	8	17	58.8	15.2	717.8	915.80	74.25	38.13
p6000.3	16132915	3060	10	20	1024.6	0.0	1566.9	738.75	253.48	17.08
p7000.1	14478676	2210	2	16	1447.6	123.5	1150.7	1196.4	1094.31	184.83
p7000.2	18249948	3170	0	6	1399.8	251.25	1481.40	1617.6	–	616.95
p7000.3	20446407	3170	20	20	0.0	0.0	296.8	215.37	109.26	50.33
p10000.1	24197906	3740	0	9	4023.9	1427.2	1719.08	2219.1	–	487.88
p10000.2	30627637	5260	0	3	4635.7	1068.85	2030.64	2584.0	–	2105.00
p10000.3	34690255	6330	0	1	3100,7	2326,85	3016.58	3253.25	–	5400,78
Среднее значение			12.1	16.0	681.9	189.3	721.3	652.8	1033.1	399.00

3. Результаты экспериментальных расчетов. В табл. 2 представлены результаты вычислительных экспериментов по решению тестовых задач вида (1) алгоритмами MST2 и ГРП.

Пусть $x^{max}(i)$ и $t^{max}(i)$ (сек) соответственно лучшее решение, найденное алгоритмом, и время его нахождения при i-й, $i = 1, \ldots, 20$, попытке решения. В табл. 2, 3 использованы следующие обозначения: $g_{avr} = f(x^{BKS}) - \frac{1}{20}\sum_{i=1}^{20} f(x^{max}(i))$; $t_{avr} = \frac{1}{20}\sum_{i=1}^{20} t^{max}(i)$; $t_{best} = \min\limits_{1 \le i \le 20} \{t^{max}(i) \mid f(x^{max}(i)) \ge f(x^{BKS})\}$; $I^{max} = \{i \mid f(x^{max}(i)) \ge f(x^{BKS})\}$; $success = |I^{max}|$ — число найденных алгоритмом решений задачи со значением целевой функции, не меньшим рекорда $f(x^{BKS})$; t_{max} — максимальное время, выделенное на решение задачи.

Из табл. 2 видно, что алгоритм ГРП по всем показателям превосходит алгоритм MST2. Отметим только полную «неудачу» алгоритма MST2 на тестовых за-

дачах при $n = 10000$, поскольку он не смог даже приблизиться к результатам алгоритма ГРП. Далее для сравнения был подключен недавно разработанный алгоритм HMA. Как отмечено в работе [4], главное достоинство этого алгоритма состоит в нахождении лучшего решения для каждой тестовой задачи. Определим важность этого показателя. В недалеком прошлом при разработке и исследовании последовательных алгоритмов оптимизации главными их характеристиками были величины g_{avr} и t_{avr}. Целесообразно было иметь алгоритм, обладающий минимальными значениями этих параметров, что, в свою очередь, требовало от него некоторой устойчивости — регулярно находить за небольшое время решения, близкие по значению целевой функции к рекордам. Иными словами, от хорошего алгоритма не требовалось находить решение x^{BKS} или иметь чрезвычайно малое время $t^{max}(i)$. В качестве примера рассмотрим результаты решения задачи p5000.1. Несмотря на то, что алгоритм MST2 не нашел известного рекорда, для него $g_{avr} = 396.9$, а для алгоритма HMA имеем $g_{avr} = 506.8$.

Таблица 3

Задача	success			t_{best}		
	HMA	MST2	GES	HMA	MST2	GES
p3000.1	20	20	20	3,63	5.88	3.99
p3000.2	20	20	20	5,52	10.84	0.51
p3000.3	17	20	20	8,58	10.84	0.62
p3000.4	20	20	19	7,78	18.22	0.66
p3000.5	15	18	20	22,60	26.13	9.91
p4000.1	20	20	20	4,99	11.04	1.57
p4000.2	17	20	20	34,40	23.18	9.24
p4000.3	19	20	20	35,40	27.11	1.96
p4000.4	18	20	20	53,10	34.02	1.37
p4000.5	12	20	20	89,70	68.03	24.13
p5000.1	4	0	6	153,20	> 1200	458.63
p5000.2	6	1	14	98,70	> 1200	35.45
p5000.3	14	17	17	364,50	60.21	30.30
p5000.4	3	1	5	789,60	> 1200	309.42
p5000.5	16	13	20	212,30	65.92	15.78
p6000.1	12	20	19	727,70	96.20	39.64
p6000.2	6	8	12	965,30	104.69	53.76
p6000.3	3	10	16	676,50	357.41	24.08
p7000.1	5	2	15	987,30	1542.98	260.61
p7000.2	2	0	5	1254,70	> 3000	869.90
p7000.3	7	20	20	1868,50	154.06	70.97
Среднее значение	12.2	12.1	15.0	398.3	438.9	105.80

С нашей точки зрения, возможен другой подход к сравнению алгоритмов, который становится все более важным в свете развития многопроцессорных комплексов. Проясним идею такого подхода на следующем примере. Пусть имеется P процессоров и следует решать тестовые задачи некоторым алгоритмом, размещая его копии [23] на каждом процессоре. Будем считать задачу решенной, если найдено ее решение со значением целевой функцией, не меньшим $f(x^{BKS})$. Если задача решена на одном из процессоров, то решение заканчивается на всех процессорах. Очевидно, что результаты такого эксперимента будут во многом аналогичны результатам, полученным при решении каждого теста P раз на одном

процессоре. Например, временем решения тестовой задачи будет соответствующее t_{best}. Отметим, что при таком подходе от алгоритма требуется хотя бы один раз решить задачу при P попытках, поэтому наиболее важными показателями для алгоритма становятся показатели *success* и t_{best}. Недостаточно лишь один раз решить задачу за приемлемое время, используя остальные попытки для диверсификации/интенсификации поиска, поскольку это меняет отношение к конструированию алгоритмов.

Авторами был протестирован свой компьютер и компьютер Pentium 2.66GHz CPU with 512M RAM, используемый в [4], тестовой программой из сайта http://www.cs.qub.ac.uk/itc2007/benchmarking/benchmark machine.zip. Установлено, что соотношение скоростей этих компьютеров составляет 1,41. Время, выданное этой программой для нашего компьютера, составило 315 сек. Далее были пересчитаны результаты табл. 2 и сведены в табл. 3 вместе с результатами из работы [4]. Из табл. 3 следует, что алгоритм ГРП превосходит алгоритмы HMA и MST2 как по показателю success, так и по времени t_{best} для большинства тестовых задач.

Таблица 4

Задача	HMA	MST2	ГРП
p3000	48.11	71.91	15.69
p4000	217.59	163.39	38.25
p5000	1618.30	–	849.58
p6000	2369.50	558.30	117.48
p7000	4110.50	–	1201.48
p10000	–	–	11194.77

В табл. 4 представлены сводные результаты по решению тестовых задач, т.е. приведено время (в секундах), затраченное на решение всех тестовых задач одной размерности. Предполагалось, что задачи решались копиями соответствующих алгоритмов на 20 процессорах с характеристиками Pentium 2.66GHz.

Анализ представленных в табл. 3, 4 экспериментальных расчетов показал, что алгоритм HMA в рамках предложенного подхода к сравнению алгоритмов хотя и нашел рекорды для всех тестовых задач, но проиграл по быстродействию не только алгоритму ГРП, но и алгоритму MST2 на сериях тестов при $n = 4000$ и $n = 6000$.

ЗАКЛЮЧЕНИЕ

Для решения задачи булева квадратичного программирования без ограничений разработан новый алгоритм, базирующийся на использовании схемы метода глобального равновесного поиска. Проведенное сравнительное исследование этого алгоритма с лучшими на данное время алгоритмами решения этой задачи показало преимущества алгоритма ГРП как по быстродействию, так и по возможности получения лучших решений.

СПИСОК ЛИТЕРАТУРЫ

1. Alidaee B., Kochenberger G., Ahmadian A. 0–1 quadratic programming approach for the optimal solution of two scheduling problems // Intern. J. of Syst. Sci. — 1994. — **25**. — P. 401–408.
2. Gallo G., Hammer P.L., Simeone B. Quadratic knapsack problems // Math. Program. — 1980. — **12**. — P. 132–149.
3. Pardalos P.M., Xue J. The maximum clique problem // J. of Global Optimiz. — 1994. — **4**. — P. 301–328.
4. Lь Z., Glover F., Hao Jin-Kao. A hybrid metaheuristic approach to solving the UBQP problem // Eur. J. of Oper. Res. — 2010. — **207**, N 3. — P. 1254–1262.

5. A unified modeling and solution framework for combinatorial optimization problems / G.A. Kochenberger, F. Glover, B. Alidaee, C. Rego // Oper. Res. Spectrum. — 2004. — **26**. — P. 237–250.
6. A polynomial case of unconstrained zero-one quadratic optimization / K. Allemand, K. Fukuda, T.M. Liebling and E. Steiner // Math. Program. Ser. A. — 2001. — **91**. — P. 49–52.
7. Pardalos P.M., Jha S. Complexity of uniqueness and local search in quadratic 0–1 programming // Oper. Res. Letters. — 1992. — **11**. — P. 119–123.
8. Helmberg C., Rendl F. Solving quadratic (0, 1) — problems by semidefinite programs and cutting planes // Math. Program. — 1998. — **82**. — P. 291–315.
9. Palubeckis G.A. Heuristic-based branch and bound algorithm for unconstrained quadratic zero-one programming // Computing. — 1995. — **54**. — P. 283–301.
10. Beasley J.E. Heuristic algorithms for the unconstrained binary quadratic programming problem. Working paper. The Management School. — London: Imperial College, 1998.
11. Glover F., Kochenberger G.A., Alidaee B. Adaptive memory tabu search for binary quadratic programs // Manag. Sci. — 1998. — **44**. — P. 336–345.
12. Palubeckis G. Iterated tabu search for the unconstrained binary quadratic optimization problem // Informatica. — 2006. — **17**, N 2. — P. 279–296.
13. Alkhamis T.M., Hasan M., Ahmed M.A. Simulated annealing for the unconstrained binary quadratic pseudo-boolean function // Eur. J. of Oper. Res. — 1998. — **108**. — P. 641–652.
14. Katayama K., Narihisa H. Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem // Eur. J. of Oper. Res. — 2001. — **134**. — P. 103–119.
15. Lodi A., Allemand K., Liebling T.M. An evolutionary heuristic for quadratic 0–1 programming // Eur. J. of Oper. Res. — 1999. — **119**. — P. 662–670.
16. Borgulya I. An evolutionary algorithm for the binary quadratic problems // Advances in Soft Computing. — 2005. — **2**. — P. 3–16.
17. Katayama K., Tani M., Narihisa H. Solving large binary quadratic programming problems by an effective genetic local search algorithm // Proc. of the Genetic and Evolutionary Computation Conference (GECCO99). — Morgan Kaufmann. — 2000. — P. 643–650.
18. Merz P., Freisleben B. Genetic algorithms for binary quadratic programming // Proceedings of the Genetic and Evolutionary Computation Conference (GECCO99). Morgan Kaufmann, 1999. — P. 417–424
19. Merz P., Katayama K. Memetic algorithms for the unconstrained binary quadratic programming problem // BioSystems. — 2004. — **78**. — P. 99–118.
20. Amini M., Alidaee B., Kochenberger G.A. A scatter search approach to unconstrained quadratic binary programs. New Methods in Optimization. — New York: McGraw-Hill, 1999. — P. 317–330.
21. Global equilibrium search applied to the unconstrained binary quadratic optimization problem / P.M. Pardalos, O.A. Prokopyev, O.V. Shylo, V.P. Shylo // Optimization Methods and Software. — 2008. — **23**. — P. 129 — 140.
22. Шило В.П. Метод глобального равновесного поиска // Кибернетика и системный анализ. — 1999. — № 1. — С. 74–81.
23. Сергиенко И.В., Шило В.П. Задачи дискретной оптимизации: проблемы, методы решения, исследования. — Киев: Наук. думка, 2003. — 264 с.

Поступила 20.06.2011