Cybernetics and Systems Analysis, Vol. 47, No. 6, November, 2011

SYSTEMS ANALYSIS

SOLVING UNCONSTRAINED BINARY QUADRATIC PROGRAMMING PROBLEM BY GLOBAL EQUILIBRIUM SEARCH

V. P. Shylo[a] **and O. V. Shylo**[b]

UDC 519.854

Abstract. *A new algorithm based on global equilibrium search (GES) is developed to solve an unconstrained binary quadratic programming (UBQP) problem. It is compared with the best methods of solving this problem. The GES algorithm is shown to be better both in speed and solution quality.*

Keywords: *binary quadratic programming, approximate methods, global equilibrium search, computational experiment, comparative analysis of algorithms.*

INTRODUCTION

We will analyze the well-known class of integer optimization problems of the form

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

where q_{ij} are elements of a symmetric real matrix Q of order n and B^n is the set of n-dimensional vectors with components 0 or 1. Such a problem is called an unconstrained binary quadratic programming (UBQP) problem. Since $x_i^2=x_i$ for all variables $x_i\in\{0,1\}$, $i=1,\dots,n$, linear function cx (if exists) can be transferred to the quadratic part of the objective function in (1). A quadratic problem of the form $\max\{xQx|Dx=b, x\in B^n\}$, where Q is the matrix from (1), $D\in R^{m\times n}$, $b\in R^n$, and R^n is the set of n-dimensional real vectors, can also be reduced to (1).

Many fundamental scientific, engineering, financial, medical, etc. problems can be formulated as binary quadratic programming problems. Quadratic functions with Boolean variables naturally arise in modeling of selections and interactions. Let us consider a set of n objects, each of which can either be selected or not. Each pair (i,j) of objects is associated with the weight q_{ij}, which measures the interaction between points i and j. If an object is selected, we assume that $x_i=1$, otherwise $x_i=0$. The sum of all interactions between the selected points, the so-called global interaction, can be presented as a quadratic Boolean function $\sum_{i=1}^{n}\sum_{j=1}^{n}q_{ij}x_ix_j$. The class of such problems was analyzed in solid state physics. Moreover, versions of problem (1) with and without constraints can be applied in numerous branches such as medicine, computer-aided design, planning, message control, chemistry [1, 2]. Many problems in graph theory can be presented in terms of binary quadratic programming, including the well-studied maximum clique and maximum cut problems [3]. Moreover, UBQP problem is a general model for a wide range of discrete optimization problems [4]. The paper [5] exemplifies its use in graph coloring, packing, splitting, linear ordering, and other problems.

The UBQP problem belongs to the class of *NP*-hard discrete optimization problems. There is a limited number of its subclasses known to be polynomially solvable (for example, [6]). It is also known that the problem "whether an UBQP

[a]V. M. Glushkov Institute of Cybernetics, National Academy of Sciences of Ukraine, Kyiv, Ukraine, v.shylo@gmail.com. [b]University of Pittsburgh, Pittsburgh, USA, *olegio@gmail.com*. Translated from Kibernetika i Sistemnyi Analiz, No. 6, pp. 68–78, November–December 2011. Original article submitted June 20, 2011.

1060-0396/11/4706-0889

problem has a unique solution?" is *NP*-hard [7]. Moreover, a binary quadratic programming problems remains *NP*-hard even if the global optimum is known to be unique [7].

Since problem (1) is *NP*-hard, exact algorithms (such as [8, 9]) are applicable only for the dimension of several hundreds of variables and moderately sparse matrix *Q*. For high-dimensional problems with dense matrices, only approximate methods are applicable, which allow finding almost optimal solutions in a reasonable time. Among those are algorithms based on tabu search [4, 10–12] and simulated annealing [11, 13, 14]. Moreover, evolutionary [15–18], memetic [19], and scatter [20] algorithms, which employ populations are well-behaved.

The paper [21] proposes the global equilibrium search (GES) algorithm for the UBQP problem and compares it with the MST2 algorithm [12], the best at that time. Though this comparison showed the advantage of the GES algorithm, it used nonstandard tests because of a program error (with erroneously zeroed diagonal elements of the matrix). Since the publication of the paper [21], new interesting algorithms (for example [4]) have been developed and we changed our approach to constructing and comparing the algorithms.

In the present paper, we will use the GES method to develop a new solution algorithm for the UBQP problem, analyze it, and compare with the well-known algorithms that use standard tests (with their dimension increased to $n = 10000$).

GES ALGORITHM

The objective function of problem (1) generates a similar landscape for different tests. It is characterized by individual peaks of approximately the same height and spaced rather far apart. If we call the domain of attraction of these peaks a mountain system, all of them belong to different mountain systems, which makes it impossible to pass from one peak to another by means of local search. This is indicative of extreme complexity of UBQP problems, especially if the optimization criterion is obligatory finding of the best known solution (preferably in the least possible time). Therefore, to create a good solution algorithm for these problems, it is necessary to take into account their special features. Let us consider a modification of the GES principal scheme [22, 23] for problem (1).

The GES principal scheme includes two stages: generating the solution and searching for the local maximum near this solution.

Assume that set S is a subset of the set of feasible solutions of problem (1) found by the GES method and

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

The "temperature" cycle is a key point in the GES structure. It starts searchings for the best solution for increasing temperature. This cycle allows flexible alternation of the diversification and intensification of the solution search domain, which finally makes the GES method highly efficient. The following should be specified for the temperature cycle: the cycle index K and the values of temperature $\mu_0 < \mu_1 < \ldots < \mu_K$ with subscripts that correspond to the numbers of the temperature cycle. We additionally define the following quantities for $k = 0, \ldots, K,\ j = 1, \ldots, n,\ u \in \{0,1\}$:

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

In the GES method, the set S is used to randomly generate the initial solutions for the search method. The vector of temperature parameters $\mu = (\mu_0, \ldots, \mu_K)$, $\mu_0 < \mu_1 < \ldots < \mu_K$, allows controlling the distance between the generated initial solutions and the best solution x_{max} in the set S. This is because the initial solution is generated by randomly changed components of the vector x_{max} by the following rule: if the jth component of the vector x_{max} is equal to zero, it varies with probability $p_j(\mu_k)$, otherwise with probability $1 - p_j(\mu_k)$. The probabilities of the proposed version of the algorithm were calculated by the formulas

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

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

It follows from (2) that E_{kj}^{u} are equal to the weighed sum of the values of the objective function over the set of known solutions whose jth components are equal to u. The weight of each solution depends on the value of its objective function and the value of the temperature parameter μ_k. A new solution is not saved and is used to recalculate the values of E_{kj}^{u}.

Let $g_j^u = \max\{f(x) | x \in S_j^u\}$, $j = 1,\ldots,n$, $u \in \{0,1\}$. As follows from (2) and (3), $\lim_{\mu_k \to \infty} p_j(\mu_k) \to 1$ as $g_j^0 < g_j^1$, otherwise $\lim_{\mu_k \to \infty} p_j(\mu_k) \to 0$. Relations (2) and (3) are obtained from the approximation of the Boltzmann distribution [21–23]. Components μ_k of the temperature vector μ are calculated from the formulas $\mu_0 = 0, \mu_{k+1} = \alpha\mu_k$, $k = 1,\ldots,K-1$.

Let us consider the scheme of the GES algorithm for the solution of a UBQP problem.

Input data: μ is the vector of temperature parameters, K is the cycle index of the temperature cycle, *maxnfail* is restart parameter, and *ngen* is the number of solutions generated in each cycle.

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. $x^{max} = \arg\max\{f(x) | x \in S\}$
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}

The main loop (rows 2–23) repeats up to the stopping criterion. In the experiments, the algorithm stopped if the running time exceeded some limiting value. As indicated above, the key element in the structure of the GES method is the temperature cycle (rows 8–17). Its index K and the vector of temperature parameters $\mu = (\mu_0,\ldots,\mu_K)$ are predetermined. The values of $\alpha > 0$ and μ_1 should be chosen so that the probability vector $p(\mu_K)$ is close to the best solution from the set S.

The probabilities with which new solutions are generated are calculated with the use of (2) and (3) in the beginning of each temperature cycle (row 9). For each probability vector, *ngen* solutions are generated (row 11). They are used as the initial solutions for the procedure Tabu search method (row 12). The set R of the solutions found by this procedure is used to supplement the set S (row 13). As stated above, the set S is used in pseudocode to simplify the description of the algorithm. In the implementation of the algorithm, it is not stored, only the values E_{kj}^{u} are.

The only solutions the algorithm stores are so-called elite solutions, which constitute the set *EliteSet* (row 22). It is assumed that due to the search intensification loop (rows 6–21), their neighborhoods are well investigated and do not contain improving solutions. Thus, the further search is performed only among the solutions for which the Hamming distance from the set *EliteSet* is no less than d_p. Therefore, the algorithm does not search in the domains already analyzed.

In the tabu search algorithm used in the implementation of the second stage of the GES algorithm, neighborhood of radius 1 is used. A solution y belongs to the neighborhood $N_1(x)$ of radius 1 centered at the point x if the Hamming distance $d(x, y)$ is equal to unity. In other words, if $y \in N_1(x)$, then $y_j = 1 - x_j$ for some j and $y_k = x_k$ for $k \neq j$. We assume that such a solution y is found with the use of $m_j : y = m_j(x)$. Let $gains(x)$ be a vector whose jth component is the increase in the objective function obtained by m_j. It can be calculated in a linear time

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

Let us now consider the scheme of the tabu search, which is used in the GES algorithm.

Input data: x is the initial solution, $g(x)$ is vector *gains*, n_{bad} is the number of iterations without improvement of the solution, tenure is a parameter, x^{max} is the best solution in the set S, x^{best} is the record found by the algorithm, and *nrep* is the cycle index of the search intensification cycle (rows 6–21) in GES Procedure.

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

Let us consider the procedure of solution generation according to the probability defined by (3).

Input data: x is the solution to which random perturbation is applied and $p(\mu_k)$ is the probability vector calculated from formulas (2) and (3).

Function:

1. $dist = 0; j = 1$
2. while $(j \le n)$ do
3. if $x_j = 1$ then
4. if $p(\mu_k) \le random[0,1]$ then
5. $x_j = 0; dist = dist + 1$
6: end if
7. else
8. if $p(\mu_k) \ge random[0,1]$ 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

Based on the vector x and with the use of the generation procedure, inversion operators are carried out (rows 2–16). The variable $random[0,1]$ uniformly distributed in (0, 1) determines the possibility of applying inversion operators to it (rows 4 and 8). The procedure is terminated when the Hamming distance between x and the perturbed solution is equal to the parameter $dist_{max}$.

COMPARATIVE ANALYSIS OF THE ALGORITHMS

Let us describe the computational experiments on using the GES algorithm to solve high-dimensional test problems (1) and compare the results with those produced by the best known algorithms.

1. Test Problems.

To carry out experimental calculations, 24 randomly generated high-dimensional ($3000 \le n \le 10000$) problems p3000.1, ..., p10000.3 with the sparsity of the matrix from 0.5 to 1 have been chosen [12]. The standard set of tests has been expanded with problems of dimension $n = 10000$ because the experience gained since the paper [21] allowed improving the efficiency of the algorithms. The codes to generate these problems are available at http://www.soften.ktu.lt/~gintaras/ubqop_its.html. It is these problems that are actively used to test the efficiency of algorithms.

Note that we do not consider here the test problems from the ORLIB library for $50 \le n \le 2500$ and similar problems from http://people.brunel.ac.uk/~mastjjb/jeb/orlib/bqpinfo.html since the GES algorithm can solve them quickly not using all of its capabilities.

2. Computing Plan. The GES algorithm was implemented in C++ language, all the computational experiments used PC Intel®Core QUAD CPU Q9550 2.83 GHz and 3.0 GB RAM. The values of the key parameters of the GES algorithm are presented in Table 1.

The initial probabilities $p_j(\mu_0) = \frac{1}{2}, j = 1,\dots,n,\ \mu_0 = 0$. The following values are used for the temperature schedule: $\mu_1 = 10^{-7},\ \mu_k = \mu_{k-1}\frac{\log 0.003/coef - \log\mu_1}{4}$ for $k = 2,\dots,K,\ coef = 1.8*10^8 / f(x^{BKS})$, where $f(x^{BKS})$ is a record known for the problem.

Note that the results of computational experiments presented below have been obtained without adjustment of the parameters, i.e., all the parameters are constant for all the test problems. The values of the parameters *tenure* and *nbad* in the tabu search algorithm from [4] confirm that it is the scheme of the GES method rather than the search techniques makes the GES algorithm so efficient.

For the comparative experimental analysis with the GES algorithm, MST2 algorithm [12] is chosen since the MST2 implementation of the tabu search algorithm is currently one of the best to solve the UBQP problem. It demonstrates good and stable operation of the algorithm for various sets of reference tests. This follows from the comparative study [12], where the results with the use of other well-known approaches are presented, including tabu search, simulated annealing, and genetic local search. Moreover, the original text of the algorithm is available at http://www.soften.ktu.lt/~gintaras/ubqop_its.html. This made it possible to use one personal computer to start the algorithms. Hence, the problem time below can be one of the parameters to compare the algorithms.

TABLE 1

Parameter	Parameter location	Meaning	Value
K	Procedure GES (row 8)	The number of iterations of the temperature cycle	6
ngen	Procedure GES (row 10)	The number of generated initial solutions	$\begin{cases}45, nrep = 0\\80, nrep > 0\end{cases}$
maxnfail	Procedure GES (row 6)	Index of the search intensification loop (rows 6–21)	1
tenure	Procedure Tabu search method (rows 6, 26)	Initial value of the tabu variable in the tabu search algorithm	*n/150*
d_p	Procedure Tabu search method (row 13), Procedure generate initial solution (row 4, 8)	The minimum admissible distance to the set of elite solutions *EliteSet*	*200*
$dist_{max}$	Procedure generate initial solution (row 12)	Maximum number of random perturbations	$\begin{cases}n, nrep = 0\\d_p, nrep > 0\end{cases}$

TABLE 2

Problem	$f(x^{BKS})$	t_{max}, sec	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
Average			12.1	16.0	681.9	189.3	721.3	652.8	1033.1	399.00

Each of the test problems was solved 20 times by each of the algorithms for various initial values of the random-number generator. The MST2 algorithm performed 1000 iterations; the time it took to solve the problem (see Table 2) was a stopping criterion for the GES algorithm. Such a plan of experiments also allows making a comparison with the promising HMA algorithm developed recently [4].

3. Results of Experimental Calculations. Table 2 summarizes the results of the computational experiments conducted to solve test problems (1) by the MST2 and GES algorithms.

TABLE 3

Problem	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
Average	12.2	12.1	15.0	398.3	438.9	105.80

Let $x^{max}(i)$ and $t^{max}(i)$ (sec) be the best solution found by the algorithm and the time it took to find it for the ith $i = 1, \ldots, 20$, solution attempt, respectively. The following notation is used in Tables 2 and 3: $g_{avr} = f(x^{BKS}) - \frac{1}{20}\sum_{i=1}^{20} f(x^{max}(i))$; $t_{avr} = \frac{1}{20}\sum_{i=1}^{20} t^{max}(i)$; $t_{best} = \min_{1 \le i \le 20} \{t^{max}(i) | f(x^{max}(i)) \ge f(x^{BKS})\}$; $I^{max} = \{i | f(x^{max}(i)) \ge f(x^{BKS})\}$; $success = |I^{max}|$ is the number of problem solutions found by the algorithm, with the value of the objective function no less than the record $f(x^{BKS})$, and t_{max} is the maximum admissible time for the problem solution.

As is seen from Table 2, the GES algorithm is better than the MST2 algorithm in all the parameters. Noteworthy is the complete failure of the MST2 algorithm with the test problems for $n = 10000$ since it cannot even approach the results of the GES algorithm. Then the HMA recently developed was used for comparison purposes. As indicated in [4], the main advantage of this algorithm is that it finds the best solution for each test problem. Let us determine how important this parameter is. Recently, the quantities g_{avr} and t_{avr} were the main characteristics of sequential optimization algorithms in their development and analysis. There was a need for an algorithm with the minimum values of these parameters, which, in turn, required it to be somehow stable, i.e., regularly and quickly find solutions with the objective function close to records. In other words, a good algorithm was not required to find a solution x^{BKS} or to have an extremely small time $t^{max}(i)$. As an example, let us consider the results of the solution of problem p5000.1. Although the MST2 algorithm has not found a known record, its $g_{avr} = 396.9$, while $g_{avr} = 506.8$ for the HMA algorithm.

We believe that there is another approach to the comparison of the algorithms, which becomes more and more important in view of the development of multiprocessor systems. Let us use the following example to clarify the idea of such an approach. Given P processors, it is required to solve test problems with some algorithm, whose copies [23] are assigned to each processor. A problem is assumed to be solved if its solution with the value the objective function no less than $f(x^{BKS})$ is found. If a problem is solved by one processor, then the solution stops on all of the processors. Obviously, the results of such an experiment will in many respects be similar to the results obtained in each test P times by one processor. For example, the time it takes to solve a test problem is the respective t_{best}. Note that such an approach requires the algorithm to solve the problem at least once in P trials; therefore, success and t_{best} become the most important parameters for the algorithm. It is insufficient to solve the problem only once in a reasonable time, using the other trials to diversify/intensify the search since this changes the view on the design of algorithms.

TABLE 4

Problem	HMA	MST2	GES
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

We used the test program machine.zip at http://www.cs.qub.ac.uk/itc2007/benchmarking/benchmark to test our computer and the computer Pentium 2.66GHz CPU with 512M RAM used in [4]. It was established that the ratio of the computer speeds is 1.41. The time for our computer was estimated as 315 sec. Then the results of Table 2 were recalculated and summarized in Table 3 together with the results from [4]. As follows from Table 3, the GES algorithm is better than the HMA and MST2 algorithms both in the *success* parameter and in the time t_{best} for the majority of test problems.

Table 4 summarizes the time (in seconds) of solving all the test problems of one dimension. The problems were supposed to be solved by the copies of the corresponding algorithms on 20 Pentium 2.66GHz processors.

An analysis of Tables 3 and 4 shows that though the HMA algorithm has found records for all the test problems within the framework of the proposed approach to the comparison of the algorithms, its speed is less than that of not only the GES algorithm but also the MST2 algorithm on a series of tests for $n = 4000$ and $n = 6000$.

CONCLUSIONS

A new algorithm based on global equilibrium search has been developed to solve an unconstrained binary quadratic programming problem. The algorithm has been compared with the best solution algorithms for this problem and has shown to be better both in speed and in the capability of finding the best solutions.

REFERENCES

1. B. Alidaee, G. Kochenberger, and A. Ahmadian, "0–1 quadratic programming approach for the optimal solution of two scheduling problems," Intern. J. Syst. Sci., **25**, 401–408 (1994).
2. G. Gallo, P. L. Hammer, and B. Simeone, "Quadratic knapsack problems," Math. Program., **12**, 132–149 (1980).
3. P. M. Pardalos and J. Xue, "The maximum clique problem," J. Global Optimiz., **4**, 301–328 (1994).
4. Z. Lь, F. Glover, and Hao Jin-Kao, "A hybrid metaheuristic approach to solving the UBQP problem," Eur. J. Oper. Res., **207**, No. 3, 1254–1262 (2010).
5. G. A. Kochenberger, F. Glover, B. Alidaee, and C. Rego, "A unified modeling and solution framework for combinatorial optimization problems," Oper. Res. Spectrum., **26**, 237–250 (2004).
6. K. Allemand, K. Fukuda, T. M. Liebling, and E. Steiner, "A polynomial case of unconstrained zero-one quadratic optimization," Math. Program., Ser. A, **91**, 49–52 (2001).
7. P. M. Pardalos and S. Jha, "Complexity of uniqueness and local search in quadratic 0–1 programming," Oper. Res. Letters, **11**, 119–123 (1992).
8. C. Helmberg and F. Rendl, "Solving quadratic (0, 1)-problems by semidefinite programs and cutting planes," Math. Program., **82**, 291–315 (1998).
9. G. A. Palubeckis, "Heuristic-based branch and bound algorithm for unconstrained quadratic zero-one programming," Computing, **54**, 283–301 (1995).
10. J. E. Beasley, "Heuristic algorithms for the unconstrained binary quadratic programming problem," Working Paper, The Management School, Imperial College, London (1998).
11. F. Glover, G. A. Kochenberger, and B. Alidaee, "Adaptive memory tabu search for binary quadratic programs," Manag. Sci., **44**, 336–345 (1998).

12. G. Palubeckis, “Iterated tabu search for the unconstrained binary quadratic optimization problem,” Informatica, **17**, No. 2, 279–296 (2006).
13. T. M. Alkhamis, M. Hasan, and M. A. Ahmed, “Simulated annealing for the unconstrained binary quadratic pseudo-boolean function,” Eur. J. Oper. Res., **108**, 641–652 (1998).
14. K. Katayama and H. Narihisa, “Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem,” Eur. J. Oper. Res., **134**, 103–119 (2001).
15. A. Lodi, K. Allemand, and T. M. Liebling, “An evolutionary heuristic for quadratic 0–1 programming,” Eur. J. Oper. Res., **119**, 662–670 (1999).
16. I. Borgulya, “An evolutionary algorithm for the binary quadratic problems,” Advances in Soft Computing, **2**, 3–16 (2005).
17. K. Katayama, M. Tani, and H. Narihisa, “Solving large binary quadratic programming problems by an effective genetic local search algorithm,” in: Proc. Genetic and Evolutionary Computation Conference (GECCO99), Morgan Kaufmann (2000), pp. 643–650.
18. P. Merz and B. Freisleben, “Genetic algorithms for binary quadratic programming,” in: Proc. Genetic and Evolutionary Computation Conference (GECCO99), Morgan Kaufmann (1999), pp. 417–424.
19. P. Merz and K. Katayama, “Memetic algorithms for the unconstrained binary quadratic programming problem,” BioSystems, **78**, 99–118 (2004).
20. M. Amini, B. Alidaee, and G. A. Kochenberger, “A scatter search approach to unconstrained quadratic binary programs,” in: New Methods in Optimization, McGraw-Hill, New York (1999), pp. 317–330.
21. P. M. Pardalos, O. A. Prokopyev, O. V. Shylo, and V. P. Shylo, “Global equilibrium search applied to the unconstrained binary quadratic optimization problem,” Optimization Methods and Software, **23**, 129–140 (2008).
22. V. P. Shilo, “The method of global equilibrium search,” Cybern. Syst. Analysis, **35**, No. 1, 68–74 (1999).
23. I. V. Sergienko and V. P. Shilo, Discrete Optimization Problems: Challenges, Solution Methods, Analysis [in Russian], Naukova Dumka, Kyiv (2003).