INFORMATICA, 2006, Vol. 17, No. 2, 279–296

Iterated Tabu Search for the Unconstrained Binary Quadratic Optimization Problem

Gintaras PALUBECKIS
Department of Practical Informatics, Kaunas University of Technology
Studentų 50, 51368 Kaunas, Lithuania
e-mail: gintaras@soften.ktu.lt

Received: January 2005

Abstract. Given a set of objects with profits (any, even negative, numbers) assigned not only to separate objects but also to pairs of them, the unconstrained binary quadratic optimization problem consists in finding a subset of objects for which the overall profit is maximized. In this paper, an iterated tabu search algorithm for solving this problem is proposed. Computational results for problem instances of size up to 7000 variables (objects) are reported and comparisons with other up-to-date heuristic methods are provided.

Key words: binary quadratic optimization, iterated tabu search, heuristics.

1. Introduction

We consider the following problem with quadratic objective function and binary variables

$$\text{maximize} \quad f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_i x_j + \sum_{i=1}^{n} c_i x_i, \tag{1}$$

$$\text{subject to} \quad x_i \in \{0, 1\}, i = 1, \ldots, n, \tag{2}$$

where c_{ij}, $i, j = 1, \ldots, n$, and c_i, $i = 1, \ldots, n$, are entries of a given $n \times n$ rational matrix C and rational n-vector c, respectively. We can assume that the main diagonal of C is zero, because, if not, we can add all nonzero diagonal entries to c at the same time replacing them with zeros in C.

In fact, we deal with a problem in which a set of objects is given and, unlike in linear optimization formulations, profits (any, even negative, numbers) are assigned not only to separate objects but also to pairs of them. The problem is to select a feasible subset of this set for which the overall profit is maximized. We focus on the most important special case (1), (2) of this general problem obtained by assuming that each subset of objects is a feasible solution. We call this special case the *unconstrained binary quadratic optimization problem* (UBQOP for short).

The UBQOP (1), (2) or sometimes (1), (2) with some constraints added (which, however, can be included into the objective function (Hansen, 1979)) arises in various contexts in many areas including solid-state physics (Barahona *et al.*, 1988; De Simone *et*

al., 1995), economics (Laughunn, 1970), computer-aided design (Barahona *et al.*, 1988; Boros and Hammer, 1991; Shih and Kuh, 1993; Jünger *et al.*, 1994), machine scheduling (Alidaee *et al.*, 1994), traffic message management (Gallo *et al.*, 1980), code-division multiple access wireless communications (Tan and Rasmussen, 2004), prediction of epileptic seizures (Iasemidis *et al.*, 2000), location (Dearing *et al.*, 1988; Hammer, 1968) and power network design (Carter, 1984). Furthermore, as reported by Kochenberger *et al.* (2004), the UBQOP can serve as a unified model for many other optimization problems. They can be put into the form (1), (2) by applying certain transformations. The list presented by Kochenberger *et al.* (2004) includes quadratic assignment, multiple knapsack, maximum clique, set partitioning, P-median, constraint satisfaction, maximum diversity, clique partitioning, task allocation, maximum cut and graph coloring problems. Applications of these problems are numerous. For example, Levin and Danieli (2005) proposed to use the multiple knapsack and morphological clique (a generalization of the maximum clique) formulations for improvement/redesign of composite systems. On the other side, the UBQOP can be considered as a member of a more general class of unconstrained binary nonlinear optimization problems. Such problems arise in a number of real-world applications, e.g., in system identification from input-output data pairs where it is required to find an optimal subset of inputs, which are necessary and sufficient for describing the system (Papadakis *et al.*, 2005). It can be expected that some ideas from the design of heuristic algorithms for the UBQOP could potentially be applied to other models in the area of binary nonlinear optimization.

The UBQOP and many of its special cases as well are NP-hard problems. Particularly, even recognizing positivity of $f(x)$ in the following very restricted case was shown to be an NP-complete problem (Palubeckis, 1995): the matrix C has zero lower triangle (including the main diagonal), each nonzero coefficient c_{ij} or c_i is equal to either 1 or -1, and each variable belongs to at most three terms of the quadratic part of f. Because of NP-hardness, the existing exact methods allow to solve instances of size less than 200 variables only. The most successful approaches include branch-and-bound methods of Barahona *et al.* (1989), Pardalos and Rodgers (1990), Billionnet and Sutter (1994), and Palubeckis (1995) and branch-and-cut method developed by Helmberg and Rendl (1998). For solving medium and large size instances of the UBQOP, only heuristic techniques are applicable. Recently, a number of iterative improvement methods have been proposed. Glover *et al.* (1998) developed a tabu search algorithm based on a flexible memory system. Their algorithm has shown good performance on a set of problems with up to 500 variables. Another tabu search algorithm was proposed by Beasley (1998). In the same paper, also an implementation of the simulated annealing metaheuristic for the UBQOP was described. Both algorithms were tested on several series of instances of size up to 2500 variables. An evolutionary heuristic for solving the UBQOP was developed by Lodi *et al.* (1999). The effectiveness of this approach has been shown on small and medium size instances. Merz and Freisleben (1999) presented a hybrid genetic algorithm incorporating a local search procedure. Using this approach, they have found new best solutions for several large Beasley problems. A scatter search algorithm for the UBQOP was proposed by Amini *et al.* (1999). Their algorithm is capable of finding solutions of good

quality even to large problem instances. Katayama and Narihisa (2001) have described a simulated annealing-based heuristic in which the annealing process is started a specified number of times from different initial temperatures. Such a multistarting mechanism improves performance of simulated annealing significantly. Computational tests were performed using problem instances with the number of variables ranging from 500 to 2500. Palubeckis and Tomkevicius (2002) developed a greedy randomized adaptive search procedure (GRASP) for solving the UBQOP. Especially good results were obtained with a GRASP enhancement in local improvement phase of which a simple tabu search implementation was used. In a recent paper, Merz and Katayama (2004) proposed a memetic algorithm with the original crossover operator and incorporated randomized k-opt local search procedure. This algorithm seems, in general, to work better than that presented by Merz and Freisleben (1999). In the paper by Palubeckis (2004), five rather different multistart tabu search strategies for the UBQOP have been described. The experiments were conducted on the two sets of test problems: largest instances taken from the OR-Library (Beasley, 1996) and randomly generated problem instances of size up to 6000 and density at least 50%. Best results were obtained in the case where construction of a new starting point for tabu search is based on applying a (fully deterministic) constructive procedure to a projection of the whole problem.

It is widely believed that for very large instances of combinatorial optimization problems, iterative improvement methods become time consuming. For large problem instances, fast constructive heuristics could be applied. One of the first such heuristics for the UBQOP was a steepest ascent algorithm presented by Palubeckis (1992). Recently, very similar heuristic, called greedy, was proposed by Merz and Freisleben (2002). Several so-called "one-pass" heuristics for solving the UBQOP were developed by Glover *et al.* (2002). They were experimentally compared against the well-known DDT method of Boros *et al.* (1989).

This paper is concerned with the development of a multistart tabu search strategy which would be very simple, in fact not much more complex than the classical random restart method and, at the same time, would be competitive with the best existing algorithms for solving the UBQOP. It has been found that these requirements are fulfilled by a strategy based on solution perturbation. The method, called iterated tabu search (ITS), can be considered as an algorithm constructed following the general schema of the iterated local search (ILS) metaheuristic (Lourenço *et al.*, 2003). In the improvement phase, one of the parts of ILS, a tabu search procedure is invoked. The perturbation is applied to a solution identified as the current solution during the tabu search at the moment of termination of this procedure. The solution is perturbed by changing the values of a subset of the variables according to the formula $x_i := 1 - x_i$, $i \in \{1, \ldots, n\}$. The manipulated variables for this operation are selected in such a way that the resulting solution would not be very far from the optimum. At each iteration, the cardinality of the subset is equal to an integer randomly and uniformly drawn from some interval. It should be noted that the term *iterated tabu search* is not entirely new in the field of optimization. Smyth, Hoos and Stützle (2003) presented an iterated robust tabu search algorithm for solving MAX-SAT, the optimization variant of the satisfiability problem arising in propositional logic.

The perturbation mechanism employed in their approach invokes a tabu search procedure tuned to execute a rather small number of iterations. This mechanism is applied to one of the two possible candidates – either a solution that entered the perturbation phase last time or a locally optimal solution found in the local search phase. As outlined above, the algorithm proposed in this paper organizes iterations of tabu search in a somewhat different way.

The rest of the paper is structured as follows. The algorithm to solve the UBQOP is presented in Section 2. The two alternative multistart tabu search strategies (described in Palubeckis (2004)) are outlined in Section 3. They are used for comparisons with the new algorithm. An empirical evaluation of the relative performance of the algorithms both on a set of problems taken from the OR-Library (Beasley, 1996) and on a set of randomly generated problem instances with up to 7000 variables appears in Section 4. Finally, in Section 5, conclusions are drawn.

2. The Algorithm

In this section, we develop an iterated tabu search algorithm for solving the unconstrained binary quadratic optimization problem. We give a detailed description of the solution perturbation procedure used to generate starting points for tabu search runs.

It has been found that it is very convenient to work with the transformed instances of (1), (2). Given a solution $x = (x_1, \ldots, x_n)$ to (1), (2), the new instance is constructed by mapping x to the zero vector. This is achieved by replacing x_i in (1) with $1 - x_i$ for each i such that $x_i = 1$. Let c'_{ij}, c'_i be the coefficients of the objective function after this operation. Then, denoting $c(i, j) = c_{ij} + c_{ji}$, we have the following relationships

$$c'_{ij} = c_{ij}(1 - 2(x_i - x_j)^2), \tag{3}$$

$$c'_i = (1 - 2x_i)(c_i + \sum_{j, x_j = 1} c(i, j)). \tag{4}$$

The constant term of the new objective function f' is equal to $f(x)$. Thus $f'(0, \ldots, 0) = f(x)$. When dealing with the transformed instance, we always drop the constant term from f'. If $c'_q > 0$ for some $q \in \{1, \ldots, n\}$, then solution x can be improved by substituting the component x_q with $1 - x_q$. For the resulting solution x', $f(x') = f(x) + c'_q$. This simple property allows to organize evaluation of solutions in the neighborhood $N_1(x) = \{(x'_1, \ldots, x'_n) \mid x'_i \in \{0, 1\}, i = 1, \ldots, n, \sum_{i=1}^{n} |x'_i - x_i| \leqslant 1\}$ of x very efficiently. The coefficients c'_i of the transformed problem also play the crucial role in the solution perturbation procedure we present later in this section. After fixing x_q at the opposite value, the coefficients c'_q and c'_i, $c'(i, q) = c'_{iq} + c'_{qi}$ for each nonzero $c(i, q)$ must be updated using the following formulas

$$c'_q := -c'_q, \tag{5}$$

$$c'_i := c'_i + c'(i, q), \tag{6}$$

$$c'(i, q) := -c'(i, q). \tag{7}$$

This operation keeps solution x' to the original problem (1), (2) in correspondence with the zero vector treated as a solution of the transformed problem instance.

The algorithm we propose for solving (1), (2) repeatedly invokes the following two procedures: GSP (Get Start Point) for construction of a starting $0-1$ vector and TS for performing tabu search. The parameter $\tilde{r}$ submitted to the first of them specifies the number of components of x whose values must be replaced by the opposite ones, that is, 0 to 1 or vice versa. The algorithm can be described as follows.

ITS

1. Randomly generate a $0-1$ vector x. Set $x^* := x$.
2. Transform the problem instance according to formulas (3), (4) applied with respect to x.
3. Call TS(x, x^*).
4. Check if a stopping criterion is satisfied. If so, then stop with the solution x^* of value $f^* := f(x^*)$. Otherwise proceed to 5.
5. Apply GSP($x, \tilde{r}$), where $\tilde{r}$ is an integer number randomly drawn from the interval $[d_1, d_2 n]$. Go to 2.

In the algorithm, the positive parameters $d_1 \leqslant n$ and $d_2 \leqslant 1$, $d_1 \leqslant d_2 n$, control the depth of solution perturbation. If d_2 is small, then less components of the vector x change their values. The stopping criterion used in Step 4 may be any. In the experiments, a stopping rule based on the CPU clock was applied.

The procedure TS invoked in Step 3 of the above algorithm is an adaptation of a simple tabu search implementation proposed by Beasley (1998). It contains only the main ingredient of tabu search, namely, a short-term memory tabu list without aspiration criterion. Due to the applied computational trick – mapping of the current solution to the zero vector – this modification is significantly faster than the original tabu search procedure described by Beasley (1998). The input to TS includes two $0-1$ vectors: an initial solution x and the best solution x^* found so far. We should note that the best vector x^* is not overridden. It is passed from one run of TS to the next one. Thus, invocations of TS are not fully independent. In the case of finding, during the run of TS, a solution x that is better than x^*, a local search procedure LS is invoked. It accepts x and returns, through the same parameter x, a locally optimal solution together with the difference f_{local} between the value of f on this solution and that on the submitted vector x. The procedure TS can be stated as follows.

TS(x, x^*)

1. Set $m := 0,\ \tilde{f} := f(x)$, tabu value $T_i := 0,\ i = 1, \ldots, n$.
2. Set $V := -\infty,\ \alpha := 0$.
3. For $k = 1, \ldots, n$ such that $T_k = 0$ do

   3.1. Increment m by 1. If $\tilde{f} + c'_k > f(x^*)$, then set $q := k,\ \alpha := 1$ and go to 4.

3.2. If $c'_k > V$, then set $V := c'_k,\ q := k$.

4. Set $x_q := 1 - x_q,\ \tilde{f} := \tilde{f} + c'_q$. Using (5)–(7), update c'_q and $c'_i,\ c'(i,q)$ for each nonzero $c(i,q)$. If $\alpha = 0$, then go to 6. Otherwise proceed to 5.
5. Apply LS$(x, f_{\mathrm{local}}, m)$ getting, possibly, improved solution x. Set $\tilde{f} := \tilde{f} + f_{\mathrm{local}}$, $x^* := x$.
6. If m is greater than or equal to a predetermined upper limit $\tilde{m}$, then return. Otherwise, decrement T_i by 1 for each positive T_i, $i \in \{1, \ldots, n\}$. Set $T_q := T$, here T is the tabu tenure value selected experimentally. Go to 2.

In the last step of TS, two parameters, T and $\tilde{m}$, are used. In a specific implementation of TS, the tabu tenure value T was fixed at 20, except the case where n is less than 80 (in which T was set to $n/4$). This constant was found to be good by Beasley (1998), and the experiments with TS have confirmed this finding. A good choice for the maximum number of iterations $\tilde{m}$ is to take $\tilde{m} = \mu n$, where μ is a tuning factor selected experimentally.

The local search procedure applied within TS is a standard routine performing an ascent from the given point to a local optimum. It consists of the following three steps.

LS$(x, f_{\mathrm{local}}, m)$

1. Set $f_{\mathrm{local}} := 0,\ \gamma := 0$.
2. For $q = 1, \ldots, n$ perform the following actions. Increment m by 1. Check whether $c'_q > 0$. If so, then set $\gamma := 1$, $x_q := 1 - x_q$, $f_{\mathrm{local}} := f_{\mathrm{local}} + c'_q$ and, using (5)–(7), update c'_q, c'_i and $c'(i,q)$ (as in Step 4 of TS).
3. If $\gamma > 0$, then set $\gamma := 0$ and repeat 2. Otherwise, return with x, f_{local} and new value of the counter m.

The tabu search is restarted from $0 - 1$ vectors delivered by the procedure GSP implementing a strategy for perturbation of a given solution. As already mentioned, besides this solution, the input to it also includes the number $\tilde{r}$ of variables to be chosen for processing (flipping of values). The task assigned to the procedure is to select variables that must undergo a flipping operation. This process is randomized. Each variable is randomly selected from the candidate list of size b. This list is constructed by including free (not selected before) variables corresponding to the largest coefficients of the linear part of the transformed problem instance. The procedure can be formally stated as follows.

GSP$(x, \tilde{r})$

1. Set $r := 0$, $I := \{1, \ldots, n\}$.
2. Form a subset $J \subset I$, $|J| = b$, such that $c'_j \geqslant c'_i$ for each $j \in J$ and each $i \in I \setminus J$ (in other words, pick the b largest coefficients c'_j among those with indices in I).
3. Randomly select $q \in J$. Remove q from I. Set $x_q := 1 - x_q$. Update c'_q and $c'_i, c'(i,q)$ for each nonzero $c(i,q)$ (as in Step 4 of TS).

4. Increment r by 1. If $r < \tilde{r}$, then go to 2. Otherwise return with the perturbed solution x.

The experiments have shown that better performance of ITS can be observed when smaller values of b are used, for example, $b \leqslant 10$. It is easy to see that, for such b, the complexity of GSP is $O(n^2)$ provided $\tilde{r}$ is proportional to n, which is the worst case. Since $\tilde{r}$ is drawn from the interval with the constant left end the execution of the procedure in many cases is much shorter.

As it can be seen from the description of ITS, each time the procedure GSP is applied to vector x that is the last evaluated solution during a run of TS. This solution usually is rather good, though on the other hand, almost surely it is worse than the best solution found thus far. Such a strategy increases a level of diversification in the search process, yet the perturbed solution remains to be of sufficiently high quality and can serve as a good starting point for the next invocation of TS. Thus, using the terminology of the iterated local search metaheuristic (Lourenço *et al.*, 2003), we can say that the acceptance criterion in the algorithm is to take the value (solution) represented by the variable x at the moment of termination of TS. Computational experience with ITS allowed us to conclude that such an acceptance criterion together with the described perturbation strategy is an efficient mechanism for restarting the search.

3. Alternative Strategies

In this section, we outline two other implementations of the multistart tabu search approach for the problem (1), (2). Later, we numerically compare the just described iterated tabu search algorithm against these alternative multistart methods.

The first of these methods is a random restart approach. It is very simple and includes the following two steps performed iteratively: random generation of a $0-1$ vector x (and, of course, transformation of the problem instance according to (3), (4) applied with respect to x); execution of TS on the generated vector x. A formal description of this algorithm, named MST1, can be found in Palubeckis (2004).

Another algorithm used for comparisons also is described in Palubeckis (2004). In this approach, initial solutions for TS are generated in two steps. In the first of them, using a certain formula, a set of variables is constructed. Each variable outside this set is forced to 0. Thus a projection of the considered instance of the UBQOP is obtained. When constructing the set, the algorithm tries to identify variables that are most prone to flip (from 0 to 1 or vice versa) the value when moving from the current solution to an optimal one. The special formula used for this purpose can be viewed as a measure of attractiveness of variables in this respect. In the second step, a steepest ascent heuristic (Palubeckis, 1992) to the obtained projection of the problem instance is applied. The basic idea of this heuristic is to perform a steepest ascent from the centre of the k-dimensional unit cube $(0.5,\ 0.5,\ \ldots,\ 0.5)$ to some its vertex ($0-1$ vector) by fixing one variable at either 0 or 1 at each step of this climb. The details of the second algorithm, named MST2,

can be found in the paper by Palubeckis (2004). It has been shown experimentally that MST2 is superior to each of the other four multistart strategies considered in that paper.

It should be emphasized that ITS differs noticeably from the algorithms MST1, ..., MST5 described in Palubeckis (2004) despite the fact that all of them are based on the tabu search schema. A salient feature of ITS is that, to enforce search diversification, it uses a solution perturbation procedure. Each of the algorithms MST1, ..., MST5 implements a search strategy different from that of ITS. In particular, MST1 is the random restart method, which continues the search simply by randomly generating a $0-1$ vector as a new starting point. The MST2 algorithm applies a constructive heuristic to certain instances of the UBQOP obtained by fixing the values of a subset of the variables. These operations are not used in ITS. The MST3 algorithm invokes a greedy randomized procedure for generating $0-1$ vectors and, therefore, bears some similarity to the GRASP method. The MST4 algorithm maintains a set of elite solutions that are used to generate good quality initial solutions for the tabu search procedure. It is the only algorithm in the group (ITS, MST1, ..., MST5), which, like evolutionary methods, works on a population rather than on a single solution at any time. Finally, the MST5 algorithm applies a problem perturbation technique, which periodically (and temporarily) modifies the problem instance, that is, coefficients c_{ij} in (1) in order to direct the search to other promising regions of the solution space. Meanwhile, ITS incorporates a solution perturbation mechanism, not problem perturbation, which makes it different from MST5.

4. Experimental Results

The main purpose of experimentation was to evaluate the performance of the described iterated tabu search algorithm and to directly compare it with two alternative multistart techniques MST1 and MST2 as well as with two most successful recent algorithms based on different metaheuristics, namely simulated annealing algorithm (SA) of Katayama and Narihisa (2001) and memetic algorithm (MA) of Merz and Katayama (2004).

The algorithms described or outlined in this paper were coded in the C programming language. The sources are publicly available at `http://www.soften.ktu.lt/~gintaras/ubqop_its.html`. Additionally, the C code implementing SA and MA was also written. The tests were carried out on a Pentium III 800 PC. As a testbed for the algorithms, the largest instances from the OR-Library (Beasley, 1996) and some of our own were considered.

After preliminary testing, the values of the ITS parameters have been fixed: $b = 5$, $d_1 = 10$, $d_2 = 0.1$. In the program implementing ITS, these values appear as some constants. Basically, the only algorithm's parameter whose value is required to be submitted to the program is μ. It has been found that μ should increase with the problem size n. Specifically, the value of μ was set to 10000 when $n \leqslant 3000$, to 12000 when $3000 < n \leqslant 5000$, and to 15000 when $n > 5000$. The same values of μ were also used in TS in the cases where TS was applied within MST1 and MST2. The experiments with MA were done using the same parameter setting as that described in Merz

and Katayama (2004). In the case of SA, the conditions of the experiments were the same as in Katayama and Narihisa (2001), except that different stopping rules were applied. Katayama and Narihisa terminated SA upon performing the annealing process twice. In the current study, on the other hand, a stopping rule based on the CPU clock was adopted. The other parameters of SA were set to the values recommended by Katayama and Narihisa (2001).

The first experiment was conducted on a set of test problems taken from the OR-Library. Only the largest problem instances available there were considered. All ten such instances, named b2500-1, ..., b2500-10, are of size 2500 and have density 10%. The smaller problems were not used because, for them, each of the tested algorithms produced solutions with the best objective function values very quickly and, therefore, no good comparison between algorithms can be provided. Each algorithm was run 25 times on each problem instance. A maximum CPU time limit of 600 seconds was set for all runs. The numerical results are summarized in Tables 1 to 3. In Table 1, the second column provides, for each instance, the best known solution value reported in recently published studies. The next columns give the number of runs of ITS, MST1, MST2, SA and MA, respectively, when a solution of value equal to the best one was produced. The results, averaged over 10 instances, are presented in the last row. The third column of Table 2 shows the difference between the best value, displayed in the second column, and the average value of 25 solutions found by ITS. The subsequent columns give this characteristic for the rest of the algorithms. Table 3 compares the average time (in seconds) taken by each of the tested algorithms to first find a solution that is best in the run.

From Tables 1 and 2, we immediately observe that the quality of solutions produced by ITS and MST2 is very similar. Both approaches, clearly, outperform the random restart method MST1. Table 3 shows that these two algorithms are also a bit faster than MST1. We should notice that the results for MST1 and MST2 in Tables 1 and 3 are slightly better

Table 1
Performance of ITS and other algorithms on the Beasley problems: success rate

Problem	Best value	ITS	MST1	MST2	SA	MA
b2500-1	1515944	25	25	25	23	20
b2500-2	1471392	23	6	25	2	1
b2500-3	1414192	24	25	24	14	7
b2500-4	1507701	25	25	25	25	25
b2500-5	1491816	25	25	25	16	12
b2500-6	1469162	25	24	25	9	4
b2500-7	1479040	25	17	25	2	3
b2500-8	1484199	25	25	25	16	21
b2500-9	1482413	25	25	25	4	1
b2500-10	1483355	25	23	25	0	1
Average		24.7	22.0	24.9	11.1	9.5

Table 2
Performance of ITS and other algorithms on the Beasley problems: average values

Problem	Best value	Solution difference (i.e., best value – heuristic solution value)				
		ITS	MST1	MST2	SA	MA
b2500-1	1515944	0	0	0	4	13
b2500-2	1471392	9	133	0	433	645
b2500-3	1414192	11	0	11	117	173
b2500-4	1507701	0	0	0	0	0
b2500-5	1491816	0	0	0	6	55
b2500-6	1469162	0	1	0	58	190
b2500-7	1479040	0	4	0	208	416
b2500-8	1484199	0	0	0	35	3
b2500-9	1482413	0	0	0	33	321
b2500-10	1483355	0	8	0	493	446
Average		2	15	1	139	226

Table 3
Performance of ITS and other algorithms on the Beasley problems: average time to the best solution in the run (in seconds)

Problem	ITS	MST1	MST2	SA	MA
b2500-1	18	14	13	225	461
b2500-2	205	281	158	334	430
b2500-3	196	91	134	319	422
b2500-4	6	8	9	120	293
b2500-5	12	7	11	305	469
b2500-6	22	48	23	283	452
b2500-7	75	168	99	387	478
b2500-8	46	26	47	293	359
b2500-9	54	77	71	340	450
b2500-10	104	161	138	351	477
Average	74	88	70	296	429

than those reported in Palubeckis (2004), primarily because they were obtained by using a faster computer. The results for MST1, especially the running times, tend to differ more significantly because the values of μ were slightly different: in the previous experiments, μ for MST1 was set to 15000, whereas now to 10000.

Comparing ITS solutions with the results obtained by SA and MA shows that ITS is definitely superior to SA and MA in terms of both solution quality and computation time. Basically, for instances in b2500 series, SA and MA perform worse than any of

the three tabu search-based algorithms considered in this section. It can be noted that the results for SA shown in Tables 1 and 2 are better than those obtained by Katayama and Narihisa (2001). The explanation of this fact is rather simple: in the experiments of Katayama and Narihisa, each run of SA took only $15 - 16$ seconds on a Sun Ultra 5/10 (UltraSPARC-IIi 440MHz), whereas, in the experiments reported here, a time limit of 600 seconds on a Pentium III 800 PC allowed to perform more iterations of SA, and therefore significantly better solutions were found. On the other hand, comparison of the results for MA gives a different picture. Merz and Katayama (2004) conducted their experiments allowing MA to create up to $150 - 190$ generations during a run, whereas the number of generations within allotted 600 seconds on a Pentium III 800 PC was less than 20. Therefore, the results (values of the objective function) presented by Merz and Katayama are considerably better than those displayed in columns for MA in Tables 1 and 2. Additionally, MA was run once with the time limit set to two hours per problem instance. The best known solutions were found for all ten instances. In particular, for three of them, namely for b2500-1, b2500-4 and b2500-8, a time period of 600 seconds was already sufficient. The time t (in seconds) taken by MA and the corresponding number of generations g for the remaining problem instances were as follows: for b2500-2, $t = 1267$, $g = 36$; for b2500-3, $t = 718$, $g = 20$; for b2500-5, $t = 634$, $g = 19$; for b2500-6, $t = 742$, $g = 21$; for b2500-7, $t = 970$, $g = 28$; for b2500-9, $t = 4655$, $g = 187$; for b2500-10, $t = 1669$, $g = 59$. These values of g are pretty comparable to generation numbers reported by Merz and Katayama (2004).

From the results presented in Tables 1 and 2, the following general conclusion can be drawn: the problem instances in the b2500 series, though being largest for (1), (2) in the OR-Library, are not sufficiently strong for state-of-the-art algorithms for the UBQOP. Therefore, ITS and alternative approaches were also tested on a set of randomly generated problems of larger size and higher density. Specifically, these problems have density from 50% to 100% and vary in size from 3000 to 7000 variables (it was not possible to try even larger problems due to insufficient computer memory to run programs implementing algorithms on them). All nonzero coefficients of the objective function are drawn uniformly at random from the interval $[-100, 100]$. The sources of the generator and input files to replicate problem instances used in the experiments can be found at `http://www.soften.ktu.lt/~gintaras/ubqop_its.html`. Each algorithm was run 5 times on each instance. The time limit for a run was set to 15, 30, 60, 90 and 150 minutes for an instance with 3000, 4000, 5000, 6000 and 7000 variables, respectively.

The results of empirical evaluation of ITS and other algorithms for larger problems are summarized in Tables 4 to 6. The first column of Table 4 gives the names identifying the problem instances. The number of variables in an instance is included into its name. The second column indicates the density of the coefficients matrix C and vector c. The third column contains, for each instance, the value of the best solution obtained from all runs of ITS, MST1, MST2, SA and MA. The fourth column shows the difference between the best value, displayed in the third column, and the value of the best solution (out of 5) found by ITS. The remaining columns give this difference for the rest of the tested

algorithms. The structure of Table 5 (respectively, Table 6) is the same as that of Table 2 (respectively, Table 3). The set of the same problem instances of size up to 6000 variables was also used in the earlier experiments conducted by Palubeckis (2004). The results obtained with MST1 and MST2 in the current study are, on the average, slightly better than those reported earlier, most of all because more time for a run of each algorithm was allotted. Moreover, in the new experiments with MST1 and MST2, slightly different values of the parameter μ were used.

The results in Tables 4 to 6 indicate that the iterated tabu search algorithm ITS is at least as good as MST2 and clearly beats the random restart method MST1. In fact, the best solutions produced by ITS and MST2 are of very similar quality. Furthermore, when comparison is based on average values, then some superiority of ITS over MST2 can be observed. Yet, none of these two algorithms dominates another algorithm in all cases. As Table 6 shows, ITS is also a faster algorithm than MST2. By analyzing the results in Tables 4 and 5, we also find that ITS is able to provide solutions of significantly higher

Table 4

Performance of ITS and other algorithms on larger problems: best values

Problem	Dens.	Best value	Solution difference (i.e., best value – heuristic solution value)				
			ITS	MST1	MST2	SA	MA
p3000-1	50	3931583	0	0	0	0	3950
p3000-2	80	5193073	0	0	0	0	342
p3000-3	80	5111533	0	357	0	0	0
p3000-4	100	5761822	0	0	0	0	1097
p3000-5	100	5675625	0	478	0	0	478
p4000-1	50	6181830	0	0	0	0	2390
p4000-2	80	7801355	0	1686	0	504	6564
p4000-3	80	7741685	0	54	0	0	5760
p4000-4	100	8711822	0	0	0	0	2359
p4000-5	100	8908979	0	0	0	0	9028
p5000-1	50	8559355	375	2691	0	1107	4647
p5000-2	80	10836019	0	0	582	582	7519
p5000-3	80	10489137	0	3277	0	354	11552
p5000-4	100	12251874	490	3341	1199	0	15955
p5000-5	100	12731803	0	5150	0	1025	6644
p6000-1	50	11384976	0	3198	0	430	9046
p6000-2	80	14333855	88	10001	0	675	21732
p6000-3	100	16132915	2729	11658	0	0	13400
p7000-1	50	14478336	0	6778	1267	2239	13365
p7000-2	80	18248297	0	7251	679	3901	18898
p7000-3	100	20446407	0	17652	0	2264	14684
Average			175	3503	177	623	8067

Table 5
Performance of ITS and other algorithms on larger problems: average values

Problem	Best value	Solution difference (i.e., best value – heuristic solution value)				
		ITS	MST1	MST2	SA	MA
p3000-1	3931583	0	667	0	0	4784
p3000-2	5193073	97	117	97	97	1198
p3000-3	5111533	344	897	287	535	3879
p3000-4	5761822	154	335	77	308	2760
p3000-5	5675625	501	1154	382	459	2982
p4000-1	6181830	0	517	0	734	3621
p4000-2	7801355	1285	3597	804	1887	7870
p4000-3	7741685	471	1465	1284	79	7218
p4000-4	8711822	438	1246	667	536	4995
p4000-5	8908979	572	2611	717	984	9567
p5000-1	8559355	646	3893	256	2130	8173
p5000-2	10836019	1068	3540	978	2101	10790
p5000-3	10489137	1266	5644	1874	2451	14663
p5000-4	12251874	1508	6270	2126	692	17744
p5000-5	12731803	835	7320	1233	1172	12996
p6000-1	11384976	57	9213	34	2248	15287
p6000-2	14333855	1709	11626	1269	2067	23632
p6000-3	16132915	3064	14958	2673	3845	19916
p7000-1	14478336	799	9281	2175	5164	18298
p7000-2	18248297	2650	11914	2163	6186	23844
p7000-3	20446407	3078	22990	7868	8978	29584
Average		978	5679	1284	2031	11610

quality than those delivered by SA and especially MA. Comparing average values (Table 5), we see that the superiority of ITS over SA is more pronounced for the largest instances, that is, containing 6000 and 7000 variables. Compared with MA, ITS found far better solutions for most of the UBQOP instances used in the second experiment. Certainly, continuing the run of MA for more generations can produce some improvements in its performance. However, it seems that, unlike in the case of b2500 series, increasing the time limit for MA several times does not help very much (at least for the largest instances). The memetic algorithm was run once on each problem of size 3000 and 7000 setting the time limit to 5 and 15 hours, respectively. The best known solutions were obtained for the following three problems (the time to solution t in seconds and the corresponding number of generations g are given in the parenthesis): p3000-1 ($t = 2635$, $g = 49$), p3000-2 ($t = 1229$, $g = 24$), p3000-4 ($t = 10318$, $g = 247$). The values f^* of the solutions found by MA for the remaining problem instances were as follows: for p3000-3, $f^* = 5110816$; for p3000-5, $f^* = 5674778$; for p7000-1, $f^* = 14473676$; for p7000-2, $f^* = 18238055$; for p7000-3, $f^* = 20428607$. The total number of generations

Table 6

Performance of ITS and other algorithms on larger problems: average time to the best solution in the run (in seconds)

Problem	ITS	MST1	MST2	SA	MA
p3000-1	228	396	106	251	726
p3000-2	212	395	97	337	809
p3000-3	327	464	271	517	590
p3000-4	519	480	559	336	722
p3000-5	462	436	255	327	638
p4000-1	215	776	436	842	1515
p4000-2	1070	785	1082	1680	1063
p4000-3	730	1011	359	1094	1106
p4000-4	845	656	624	1002	1373
p4000-5	797	862	700	1279	1287
p5000-1	1520	2260	1621	1816	3000
p5000-2	1264	1984	1946	2072	2562
p5000-3	2015	1410	2365	2836	2925
p5000-4	1787	2005	2805	3178	2075
p5000-5	1652	1922	2156	3171	3095
p6000-1	2935	2860	3112	1844	4009
p6000-2	2517	3119	2661	3256	3688
p6000-3	2871	3217	3655	4422	4364
p7000-1	5313	4954	4348	5806	7942
p7000-2	3039	4484	5165	5215	5525
p7000-3	4339	2801	6342	6417	8197
Average	1650	1775	1936	2271	2724

within the allotted time period was about 380 and 155 for p3000 and p7000 instances, respectively.

Table 6 shows that ITS takes on the average less time to first find the best solution in a run than any of the other tested algorithms. Notice that the memetic algorithm MA is the slowest of them all.

In the final experiment, longer runs of ITS on the problems with 5000, 6000 and 7000 variables were performed. The time limit was set to 5, 8 and 10 hours, respectively. Table 7 displays the results only for those problem instances for which a solution better than that reported in Table 4 for ITS was produced. Comparing these tables, we see that ITS was able to match the solutions found by MST2 for p5000-1 and p6000-3 but, however, failed for p6000-2. Furthermore, ITS, when allowed to run longer, improved the best solutions also for p5000-4, p7000-1 and p7000-2. However, in most cases, better solutions were obtained at the cost of a significant increase in computation time.

In closing this section, we note that, for many problems in the second series, the solutions of the best-reported value were found many times with different algorithms.

Table 7
Results of longer runs of ITS

Problem	Solution value	Time (in seconds)
p5000-1	8559355	3457
p5000-4	12252318	12605
p6000-3	16132915	9830
p7000-1	14478676	30198
p7000-2	18249948	1877

Therefore, it could be argued that for a subset of instances the best values given in the paper (third column of Table 4) are very close to the optimum.

5. Conclusions

This paper described an iterated tabu search algorithm for the unconstrained binary quadratic optimization problem. The algorithm is rather simple and easy to implement. The experiments have shown that, at the same time, it is very competitive with existing state-of-the-art algorithms for the UBQOP. The proposed algorithm is reliable in the sense that it is able to produce better solutions than other tested algorithms for both sparse and dense problem instances. Moreover, this algorithm is fastest among the five considered algorithms. In particular, it required about 27% and 39% less time to find the best solutions for large UBQOP instances than the simulated annealing and, respectively, evolutionary search implementations. Thus, we can conclude that the proposed iterated tabu search algorithm performs remarkably well, both in terms of solution quality and computational speed. An open problem is a more systematic investigation of an optimal trade-off between the number of tabu search restarts and the number of iterations of a tabu search run, assuming that the time interval allotted for the execution of the whole program is fixed.

We believe that a similar solution strategy could be successfully applied to develop heuristic algorithms for some other hard combinatorial optimization problems.

References

Alidaee, B., G. Kochenberger and A. Ahmadian (1994). 0-1 quadratic programming approach for the optimal solution of two scheduling problems. *International Journal of Systems Science*, **25**, 401–408.

Amini, M.M., B. Alidaee and G.A. Kochenberger (1999). A scatter search approach to unconstrained quadratic binary programs. In D. Corne, M. Dorigo, and F. Glover (Eds.) *New Ideas in Optimization*. McGraw-Hill, London, England. pp. 317–329.

Barahona, F., M. Grötschel, M. Jünger and G. Reinelt (1988). An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, **36**, 493–513.

Barahona, F., M. Jünger and G. Reinelt (1989). Experiments in quadratic 0-1 programming. *Mathematical Programming*, **44**, 127–137.

Beasley, J.E. (1996). Obtaining test problems via Internet. *Journal of Global Optimization*, **8**, 429–433.

Beasley, J.E. (1998). Heuristic algorithms for the unconstrained binary quadratic programming problem. *Working Paper*, The Management School, Imperial College, London, England.

Billionnet, A., and A. Sutter (1994). Minimization of a quadratic pseudo-Boolean function. *European Journal of Operational Research*, **78**, 106–115.

Boros, E., and P.L. Hammer (1991). The max-cut problem and quadratic $0-1$ optimization: polyhedral aspects, relaxations and bounds. *Annals of Operations Research*, **33**, 151–180.

Boros, E., P.L. Hammer and X. Sun (1989). The DDT method for quadratic $0-1$ minimization. *RUTCOR Research Report 39–89*, Rutgers University, New Brunswick, USA.

Carter, M.W. (1984). The indefinite zero-one quadratic problem. *Discrete Applied Mathematics*, **7**, 23–44.

Dearing, P.M., P.L. Hammer and B. Simeone (1988). Boolean and graph-theoretic formulations of the simple plant location problem. *RUTCOR Research Report 3–88*, Rutgers University, New Brunswick, USA.

De Simone, C., M. Diehl, M. Jünger, P. Mutzel, G. Reinelt and G. Rinaldi (1995). Exact ground states of Ising spin glasses: new experimental results with a branch and cut algorithm. *Journal of Statistical Physics*, **80**, 487–496.

Gallo, G., P.L. Hammer and B. Simeone (1980). Quadratic knapsack problems. *Mathematical Programming*, **12**, 132–149.

Glover, F., B. Alidaee, C. Rego and G. Kochenberger (2002). One-pass heuristics for large-scale unconstrained binary quadratic problems. *European Journal of Operational Research*, **137**, 272–287.

Glover, F., G.A. Kochenberger and B. Alidaee (1998). Adaptive memory tabu search for binary quadratic programs. *Management Science*, **44**, 336–345.

Hammer, P.L. (1968). Plant location – a pseudo-Boolean approach. *Israel Journal of Technology*, **6**, 330–332.

Hansen, P. (1979). Methods of nonlinear 0-1 programming. *Annals of Discrete Mathematics*, **5**, 53–70.

Helmberg, C., and F. Rendl (1998). Solving quadratic $(0,1)$-problems by semidefinite programs and cutting planes. *Mathematical Programming*, **82**, 291–315.

Iasemidis, L.D., D.S. Shiau, J.C. Sackellares and P. Pardalos (2000). Transition to epileptic seizures: optimization. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, **55**, 55–73.

Jünger, M., A. Martin, G. Reinelt and R. Weismantel (1994). Quadratic $0/1$ optimization and a decomposition approach for the placement of electronic circuits. *Mathematical Programming*, **63**, 257–279.

Katayama, K., and H. Narihisa (2001). Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem. *European Journal of Operational Research*, **134**, 103–119.

Kochenberger, G.A., F. Glover, B. Alidaee and C. Rego (2004). A unified modeling and solution framework for combinatorial optimization problems. *OR Spectrum*, **26**, 237–250.

Laughunn, D.J. (1970). Quadratic binary programming. *Operations Research*, **14**, 454–461.

Levin, M.Sh., and M.A. Danieli (2005). Hierarchical decision making framework for evaluation and improvement of composite systems (example for building). *Informatica*, **16**, 213–240.

Lodi, A., K. Allemand and T.M. Liebling (1999). An evolutionary heuristic for quadratic 0-1 programming. *European Journal of Operational Research*, **119**, 662–670.

Lourenço, H.R., O.C. Martin and T. Stützle (2003). Iterated local search. In F.W. Glover and G.A. Kochenberger (Eds.) *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston, USA. pp. 321–353.

Merz, P., and B. Freisleben (1999). Genetic algorithms for binary quadratic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Vol. 1. Morgan Kaufmann, Orlando, Florida, USA. pp. 417–424.

Merz, P., and B. Freisleben (2002). Greedy and local search heuristics for unconstrained binary quadratic programming. *Journal of Heuristics*, **8**, 197–213.

Merz, P., and K. Katayama (2004). Memetic algorithms for the unconstrained binary quadratic programming problem. *Biosystems*, **78**, 99–118.

Palubeckis, G. (1992). Heuristics with a worst-case bound for unconstrained quadratic 0-1 programming. *Informatica*, **3**, 225–240.

Palubeckis, G. (1995). A heuristic-based branch and bound algorithm for unconstrained quadratic zero-one programming. *Computing*, **54**, 283–301.

Palubeckis, G. (2004). Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. *Annals of Operations Research*, **131**, 259–282.

Palubeckis, G., and A. Tomkevicius (2002). GRASP implementations for the unconstrained binary quadratic optimization problem. *Information Technology and Control*, **3**, 14–20.

Papadakis, S.E., P. Tzionas, V.G. Kaburlasos and J.B. Theocharis (2005). A genetic based approach to the type I structure identification problem. *Informatica*, **16**, 365–382.

Pardalos, P.M., and G.P. Rodgers (1990). Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing*, **45**, 131–144.

Shih, M., and E.S. Kuh (1993). Quadratic Boolean programming for performance-driven system partitioning. In *Proceedings of the 30th ACM/IEEE Design Automation Conference*. Dallas, Texas, USA. pp. 761–765.

Smyth, K., H.H. Hoos and T. Stützle (2003). Iterated robust tabu search for MAX-SAT. In Y. Xiang and B. Chaib-draa (Eds.) *Advances in Artificial Intelligence: Proceedings of the 16th Conference of the Canadian Society for Computational Studies of Intelligence, AI2003*. Halifax, Canada, *Lecture Notes in Artificial Intelligence*, **2671**, 129–144.

Tan, P.H., and L.K. Rasmussen (2004). Multiuser detection in CDMA — a comparison of relaxations, exact, and heuristic search methods. *IEEE Transactions on Wireless Communications* (forthcoming).

G. Palubeckis received the degree of doctor of sciences from the Kaunas Polytechnic Institute, Kaunas, Lithuania, in 1987. Currently, he is a professor at the Kaunas University of Technology. His major research interests are in graph theory, combinatorial optimization, and computational geometry.

Iteracinė tabu paieška neapribotos binarinės kvadratinės optimizacijos uždaviniui

Gintaras PALUBECKIS

Straipsnyje nagrinėjama situacija kai yra duota aibė objektų ir tam tikri dydžiai (bet kokie, netgi neigiami skaičiai), įvertinantys naudingumą, yra priskirti ne tik pavieniams objektams bet ir jų poroms. Neapribotos binarinės kvadratinės optimizacijos uždavinys reikalauja surasti objektų poaibį, kuriam suminis naudingumas būtų maksimalus. Prie tokio modelio susiveda daug praktinių uždavinių, iškylančių įvairiose srityse. Straipsnyje siūlomas iteracinės tabu paieškos algoritmas suformuluotam uždaviniui spręsti. Pateikiami skaičiavimų rezultatai uždavinio pavyzdžiams, turintiems iki 7000 kintamųjų (objektų). Algoritmas palyginamas su kitais šiuolaikiškais euristiniais metodais.