Probabilistic GRASP-Tabu Search Algorithms for the UBQP problem

Yang Wang [a], Zhipeng Lü [b], Fred Glover [c] Jin-Kao Hao [a,*]

[a] *LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

[b] *School of Computer Science and Technology, Huazhong University of Science and Technology, 430074 Wuhan, China*

[c] *OptTek Systems, Inc., 2241 17th Street Boulder, CO 80302, USA*

Accepted to *Computers and Operations Research*, Dec. 2011.
DOI: 10.1016/j.cor.2011.12.006

Abstract

This paper presents two algorithms combining GRASP and Tabu Search for solving the Unconstrained Binary Quadratic Programming (UBQP) problem. We first propose a simple GRASP-Tabu Search algorithm working with a single solution and then reinforce it by introducing a population management strategy. Both algorithms are based on a dedicated randomized greedy construction heuristic and a tabu search procedure. We show extensive computational results on two sets of 31 large random UBQP instances and one set of 54 structured instances derived from the MaxCut problem. Comparisons with state-of-the-art algorithms demonstrate the efficacy of our proposed algorithms in terms of both solution quality and computational efficiency. It is noteworthy that the reinforced GRASP-Tabu Search algorithm is able to improve the previous best known results for 19 MaxCut instances.
Keywords: GRASP; Tabu Search; UBQP; Path Relinking; Population Management; MaxCut

1 Introduction

The objective of the unconstrained binary quadratic programming problem is to maximize the function:

* Corresponding author.
Email addresses: yangw@info.univ-angers.fr (Yang Wang), zhipeng.lv@hust.edu.cn (Zhipeng Lü), glover@opttek.com (Fred Glover), hao@info.univ-angers.fr (Jin-Kao Hao).

$$f(x) = x'Qx = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j \tag{1}$$

where $Q = (q_{ij})$ is an $n \times n$ matrix of constants and x is an n-vector of binary (zero-one) variables, i.e., $x_i \in \{0, 1\},\ i = 1, \ldots, n$.

The UBQP is notable for its ability to formulate a wide range of important problems, including those from financial analysis [23], social psychology [16], machine scheduling [1], computer aided design [20] and cellular radio channel allocation [9]. Besides, due to the ability to incorporate quadratic infeasibility constraints into the objective function in an explicit manner, UBQP enables itself to serve as a common model for a wide range of combinatorial optimization problems. A review of additional applications and the re-formulation procedures can be found in [19] demonstrating the utility of UBQP for a variety of applications.

During the last two decades, a large number of procedures for solving the UBQP have been reported in the literature. Among them are several exact methods using branch and bound or branch and cut (see, e.g., [6,17,30]). Due to the fact that the exact methods become prohibitively expensive to apply for solving large instances, various metaheuristic algorithms have been extensively used to find high-quality solutions to large UBQP instances in an acceptable time. Some representative metaheuristic methods include local search heuristics [7], Simulated Annealing [4,18]; adaptive memory approaches based on Tabu Search [14,15,27,29]; population-based approaches such as Evolutionary Algorithms [5,21,25], Scatter Search [2] and Memetic Algorithms [22,26].

This paper presents two algorithms for the UBQP that combine GRASP and Tabu Search. The first, GRASP-TS, uses a basic GRASP algorithm with single solution search while the other, GRASP-TS/PM, launches each tabu search by introducing a population management strategy based on an elite reference set. In GRASP-TS/PM we pick a single solution at a time from the reference set, and operate on it, utilizing the ideas of "elite solution recovery" and "probabilistic evaluation" proposed in [12,37]. Our experimental testing discloses that GRASP-TS/PM yields very competitive outcomes on a large range of both random and structured problem instances.

To assess the performance and the competitiveness of our algorithms in terms of both solution quality and efficiency, we provide computational results on 31 large random benchmark instances with up to 7000 variables as well as 54 instances derived from the MaxCut problem.

The remaining part of the paper is organized as follows. Sections 2 and 3 describe respectively the basic GRASP-Tabu Search algorithm and the GRASP-Tabu Search algorithm with Population Management. Section 4 is dedicated to the computational results and detailed comparisons with other state-of-the-art algorithms in the literature. Finally, concluding remarks are given in Section 5.

2 GRASP-Tabu Search

2.1 General GRASP-TS procedure

The GRASP algorithm is usually implemented as a multistart procedure [31,32], consisting of a randomized greedy solution construction phase and a sequel local search phase to optimize the objective function as far as possible. These two phases are carried out iteratively until a stopping condition is satisfied.

Our basic GRASP-Tabu Search algorithm (denoted by GRASP-TS) for the UBQP follows this general scheme (see Algorithm 1) and uses a dedicated greedy heuristic for solution construction (see Section 2.2) as well as tabu search (see Section 2.3) as its local optimizer.

Algorithm 1 Pseudo-code of GRASP-TS for UBQP

Input: matrix Q
Output: the best binary n-vector x^* found so far and its objective value f^*
$f^* = -\infty$
repeat
  Construct a greedy randomized solution x^0 /∗ Section 2.2 ∗/
  $x' \leftarrow$ Tabu_Search(x^0) /∗ Section 2.3 ∗/
  if $f(x') > f^*$ **then**
    $x^* = x'$, $f^* = f(x')$
  end if
until a stopping criterion is met

2.2 Solution Construction

GRASP-TS constructs a new solution at each step according to a greedy random construction heuristic, which was originally used in probabilistic Tabu Search (PTS) [12,36,37] and motivated by the fact that the GRASP construction resembles this PTS approach.

The construction procedure consists of two phases: one is to adaptively and iteratively select some variables to receive the value 1; the other is to assign the value 0 to the left variables. Starting with an empty solution, a variable x_i with the maximum q_{ii} is picked as the first element of the partial solution.

Given the partial solution $px = \{x_{k_1}, x_{k_2}, ..., x_{k_\alpha}\}$, indexed by $pi = \{k_1, k_2, ..., k_\alpha\}$, we calculate its objective function value (OFV) as:

$$OFV(px) = \sum_{i \in pi, x_i = 1} (q_{ii} + \sum_{j \in pi, j \neq i} q_{ij} \cdot x_j) \tag{2}$$

At each iteration of the first phase we choose one unassigned variable according to a greedy function and then assign value 1 to it. Specifically, we calculate the objective function increment (OFI) to the partial solution px if one unassigned variable x_j $(j \in N \setminus pi)$ is added into px with value 1.

$$OFI_j(px) = q_{jj} + \sum_{i \in pi} (q_{ij} \cdot x_i) \tag{3}$$

At each step, all the unassigned variables are sorted in an non-increasing order according to their OFI values. Note that we only consider the first rcl variables having non-negative OFI values, where rcl is called the restricted candidate list size. The r-th ranked variable is associated with a bias $b_r = 1/e^r$. Therefore, the r-th ranked variable is selected with probability $p(r)$ that is calculated as follows:

$$p(r) = b_r / \sum_{j=1}^{rcl} b_j \tag{4}$$

Once a variable x_j is selected and added into the partial solution px with the assignment value 1, the partial solution px and its index pi, its objective function value $OFV(px)$ and the left variables' OFI values are updated correspondingly as follows:

$$px' = px \cup \{x_j\}, \ pi' = pi \cup \{j\} \tag{5}$$

$$OFV(px') = OFV(px) + OFI_j(px) \tag{6}$$

For any variable x_k $(k \in N \setminus pi')$,

$$OFI_k(px') = OFI_k(px) + q_{jk} \tag{7}$$

This procedure repeats until all the OFI values of the unassigned variables are negative. Then, the new solution is completed by assigning the value 0 to all the left variables.

2.3 Tabu Search Procedure

When a new solution is fully constructed, we apply the tabu search procedure described in [22] to optimize this solution. Our TS algorithm is based on a simple *one-flip move* neighborhood, which consists of changing (flipping) the value of a single variable x_i to its complementary value $1 - x_i$. Each time a move is carried out, the reverse move is forbidden for the next $TabuTenure$ iterations. In our implementation, we selected to set the tabu tenure by the assignment $TabuTenure(i) = ttc + rand(10)$, where ttc is a given constant and rand(10) takes a random value from 1 to 10. Once a move is performed, one needs just to update a subset of move values affected by the move. Accompanying this rule, a simple aspiration criterion is applied that permits a move to be selected in spite of being tabu if it leads to a solution better than the current best solution. Our TS method stops when the best solution cannot be improved within a given number μ of moves and we call this number the *improvement cutoff*. Interested readers are referred to [22] for more details.

3 GRASP-Tabu Search with Population Management

3.1 General GRASP-TS/PM procedure

Starting from the basic GRASP-TS algorithm, we introduce additional enhancements using the idea of maintaining a pool of elite solutions. By combining GRASP-TS with the population management strategy, our reinforced GRASP-TS/PM algorithm offers a better balance between intensification and diversification as a means of exploiting the search space. The general architecture of the GRASP-TS/PM algorithm is described in Algorithm 2, which is composed of four main components: a reference set construction procedure (lines 4, 23 in Algorithm 2, Section 3.2), a randomized greedy solution reconstruction operator (line 11 in Algorithm 2, Section 3.3), a tabu search procedure (line 12 in Algorithm 2, Section 2.3) and a reference set updating rule (lines 16-21 in Algorithm 2, Section 3.4).

GRASP-TS/PM starts from an initial reference set ($RefSet$) consisting of b local optimum solutions (line 4), from which the worst solution x^w in terms of the objective value is identified (line 6). Then, $Examine(x) = True$ indicates that solution x is to be examined (line 7). At each GRASP-TS/PM iteration, one solution x^0 is randomly chosen from the solutions to be examined in $RefSet$ ($Examine(x^0) = True$), reconstructed according to the randomized greedy heuristic and optimized by the tabu search procedure to local optimality (lines 9–12). If the improved solution x' meets the criterion of updating

$RefSet$, the worst solution x^w is replaced by x' in $RefSet$ and $Examine(x')$ is set to be $True$ (lines 16-19). Then, the new worst solution x^w is identified (line 20). This procedure repeats until all the solutions in $RefSet$ have been examined. When this happens, $RefSet$ is rebuilt as the initial reference set construction except that the best solution x^* becomes a member of the new $RefSet$ (line 23).

Algorithm 2 Pseudo-code of GRASP-TS/PM for UBQP

1: **Input**: matrix Q
2: **Output**: the best binary n-vector x^* found so far and its objective value f^*
3: $f^* = -\infty$
4: $RefSet \leftarrow$ Initialize_RefSet( ) /* Section 3.2 */
5: **while** The stopping criterion is not satisfied **do**
6:   Find the worst solution x^w in $RefSet$ in terms of the objective value
7:   Let $Examine(x^i) = True$, $i = 1, \ldots, b$ ($|RefSet| = b$)
8:   **repeat**
9:     Randomly choose one individual x^0 from $RefSet$ with $Examine(x^0) = True$
10:     $Examine(x^0) = False$
11:     $x' \leftarrow$ Reconstruct_Solution(x^0) /* Section 3.3 */
12:     $x' \leftarrow$ Tabu_Search(x') /* Section 2.3 */
13:     **if** $f(x') > f^*$ **then**
14:       $x^* = x'$, $f^* = f(x')$
15:     **end if**
16:     $UpdateSucc \leftarrow$ Update_RefSet($RefSet, x'$) /* Section 3.4 */
17:     **if** $UpdateSucc$ is TRUE **then**
18:       $RefSet \leftarrow RefSet \cup \{x'\} \setminus \{x^w\}$
19:       $Examine(x') = True$
20:       Record the new worst solution x^w in $RefSet$
21:     **end if**
22:   **until** ($\forall x \in RefSet$, $Examine(x) = False$)
23:   $RefSet \leftarrow$ Reconstruct_RefSet($RefSet$) /* Section 3.2 */
24: **end while**

3.2 RefSet Initialization and Reconstruction

The initial reference set contains b different local optimum solutions and is constructed as follows. Starting from scratch, we randomly generate a solution, improve it to local optimality by our tabu search procedure (Section 2.3) and then add it into the reference set if this solution does not occur in $RefSet$. The procedure repeats until the size of $RefSet$ reaches b.

As shown in Algorithm 2, the reference set is recreated when all the solutions in $RefSet$ have been examined. In this case, the best solution x^* becomes a member of the new $RefSet$ and the remaining solutions are generated in the same way as in constructing the initial $RefSet$.

The initial or the renewed reference set can also be obtained by applying the randomized greedy construction heuristic described in Section 2.2. However, experimental studies showed although there are no significant performance differences, random generation generally leads to slightly better results. For this reason, we adopt random generation of reference sets in this paper.

3.3 Solution Reconstruction

In GRASP-TS/PM, a new solution is reconstructed based on an elite solution, borrowing the idea of elite solution recovery strategy described in [12,37]. More specifically, GRASP-TS/PM creates a new solution by first inheriting parts of the "good" assignments of one elite solution in $RefSet$ to form a partial solution and then completing the remaining parts as GRASP-TS does. We describe how the partial elite assignments are inherited as follows.

Given an elite solution x in $RefSet$, we reconstruct a new solution by the strategic oscillation, which was proposed in the early literature [11] in a multi-start role to replace the customary multi-start design by using a destructive/constructive process that dismantles only part of a selected solution and rebuilds the remaining portion. Specifically, it exploits critical variables identified as *strongly determined*, and has come to be one of the basic strategies associated with tabu search. This idea has also been used in our recent work [34].

Let $x = \{x_1, x_2, ..., x_n\}$, indexed by $N = \{1, \ldots, n\}$. The *objective function contribution* of a given variable x_i relative to x is defined as:

$$VC_i(x) = (1 - 2x_i)(q_{ii} + \sum_{j \in N \setminus \{i\}} q_{ij}x_j) \tag{8}$$

As noted in [14] and in a more general context in [15], $VC_i(x)$ identifies the change in $f(x)$ that results from changing the value of x_i to 1 - x_i; i.e.,

$$VC_i(x) = f(x') - f(x) \tag{9}$$

where $x'_j = x_j$ for $j \in N \setminus \{i\}$ and $x'_i = 1 - x_i$. We observe that under a maximization objective if x is a locally optimal solution, as will typically be the case when we select x to be a high quality solution, then $VC_i(x) \leq 0$ for all $i \in N$, and the current assignment of x_i will be more strongly determined as $VC_i(x)$ is "more negative".

After calculating each variable's VC value, we sort all variables in a non-decreasing order according to their VC values. Then the top α variables are selected and assigned the same values as in x. Thus, the assignments of these α strongly determined variables form a partial solution. Note that, instead of using the "strongly determined" move evaluations described above, an alternative way to make the probabilistic assignments can be based on the "consistent variables" evaluations given by the population of elite solutions as shown in [11]. In addition, a combination of the population-based determination and the move value-based determination would also be possible, as shown in [35].

With the partial elite solution, we fix the remaining variables of the new solution using the randomized greedy heuristic as in GRASP-TS (see Section 2.2). Note that GRASP-TS starts with an empty solution to construct an initial solution.

3.4 RefSet Updating

The updating procedure of $RefSet$ is invoked each time a newly constructed solution is improved by tabu search. Specifically, the improved solution is added into $RefSet$ if it is distinct from any solution in the $RefSet$ and better than the worst solution x^w in $RefSet$ in terms of the objective function value. Under this circumstance, x^w is replaced and thus $RefSet$ is updated.

3.5 Relations between GRASP-TS/PM and HMA [22]

The proposed GRASP-TS/PM algorithm shares some similarities with the leading HMA algorithm [22] in the sense that both algorithms manage a pool of solutions and use tabu search as their local optimization procedure. However, there are notable differences between them concerning the other key components.

First, GRASP-TS/PM uses a dedicated method to reconstruct, from one elite solution, a new solution with a randomized greedy heuristic while HMA recombines two solutions with two crossover operators. Second, HMA updates its population by considering both quality and distance while the GRASP-TS/PM uses a simpler rule by considering only the quality criterion. Third, GRASP-TS/PM and HMA employ different rules to generate the initial population. Fourth, GRASP-TS/PM renews its population once each of its solutions has been used for reconstruction while HMA has no corresponding operation. In summary, the proposed algorithm is simpler than HMA in its design and implementation. Yet, as we see below, GRASP-TS/PM is able to achieve a very competitive performance.

Table 1
Settings of Important Parameters

Parameters	Section	Description	Values	
			UBQP	MaxCut
b	3.2	RefSet size	10	10
α	3.3	elite inheritance variables	$0.25 \cdot n$	$0.25 \cdot n$
rcl	2.2	restricted candidate list	50	50
ttc	2.3	tabu tenure constant	$n/100$	$n/10$
μ	2.3	improvement cutoff of TS	$5 \cdot n$	10000

4 Computational Results

4.1 Test Instances

Three sets of test problems are considered in the experiments. Two of them are random UBQP problems and the other one is derived from the MaxCut problem. The two sets of random UBQP benchmarks are composed of 10 instances with size of 2500 from ORLIB [3] and 21 larger instances with size ranging from $n = 3000$ to 7000 from http://www.soften.ktu.lt/∼gintaras/ubqop_its.html. Experiments reported in [15,22,27,29] showed that the large instances with more than 5000 variables are particularly challenging.

The MaxCut benchmarks used contain 54 instances named G1,. . .,G54 with size ranging from $n = 800$ to 3000 which are available at http://www.stanford.edu/∼yyye/yyye/Gset. These instances are created by using a machine-independent graph generator, comprising of toroidal, planar and random weighted graphs with weight values 1, 0 or -1. Many authors including [8,10,24,28,33] employ these instances to test their algorithms. Note that we use the UBQP model to solve the MaxCut problem through a simple transformation according to [19].

4.2 Experimental Protocol and Parameter Setting

Our GRASP-Tabu Search algorithms are programmed in C and compiled using GNU GCC on a PC running Windows XP with Pentium 2.83GHz CPU and 2GB RAM. All computational results were obtained without special tuning of the parameters, i.e., all the parameters used in our algorithm are fixed (constant) for all instances considered. Table 1 gives the descriptions and settings of the parameters used in the two proposed algorithms, where the last two columns respectively denote the settings for the set of 31 random UBQP instances and the set of 54 MaxCut instances.

These parameter values were determined based on preliminary experiments. For instance, we experimented with selecting $rcl \in \{50, 0.1 \cdot n, 0.2 \cdot n, 0.3 \cdot n,$

$0.4 \cdot n$, $0.5 \cdot n$, $1.0 \cdot n\}$ on a preliminary set of problem instances and observed that $rcl = 50$ is a good compromise in terms of the best objective value, average average objective value, standard deviation and CPU time. The size of RefSet (parameter b) was fixed similarly. Better parameter values would be possible in some cases, but as we see below, the proposed algorithms with the given parameter values are able to achieve a highly competitive performance.

Our GRASP-TS algorithm uses the CPU clocks as the stop condition while the GRASP-TS/PM algorithm requires the completion of at least one round of the GRASP-TS/PM process. The time limit for the 10 ORLIB instances for a single run is set to be 1 minute and for the 21 larger random instances with 3000, 4000, 5000, 6000 and 7000 variables is 5, 10, 20, 30 and 50 minutes, respectively. Note that this time cutoff is the same as in [22]. In addition, we set 30 minutes as the stop condition for the 54 MaxCut instances, which is comparable with the time reported in [24].

Given the stochastic nature of our GRASP-Tabu Search algorithms, we solve each problem instance independently 20 times and show statistics over these 20 runs.

4.3 Computational Results on the Random UBQP Instances

Table 2 shows the computational statistics of the GRASP-TS and GRASP-TS/PM algorithms on the 31 UBQP instances. Columns 1 and 2 respectively give the instances names and the best known objective values f_{prev} in the literature. Note that these best values were first reported in [27,29] and recently improved in [15,22]. The columns under heading "GRASP-TS" and "GRASP-TS/PM" list the best objective value f_{best}, the average objective value f_{avr}, the standard variance of the objective value σ and the average CPU time *time* (seconds) for reaching f_{best} over the 20 runs. Furthermore, the last row "Average" indicates the summary of average performances of our algorithms.

Table 2 discloses that generally GRASP-TS/PM performs better than GRASP-TS on these UBQP benchmarks. First, we notice that both GRASP-TS and GRASP-TS/PM can reach all the previous best objective values for the 31 UBQP instances within the given time limit, demonstrating their very good performance in finding the best solution. However, GRASP-TS/PM is superior to GRASP-TS when it comes to the average gap to the previous best objective values g_{avr} on these instances, 316.9 versus 509.6, although the CPU time to obtain the best solution is slightly longer. Moreover, the average variance of GRASP-TS/PM is 252.0, which is much smaller than 386.4 of GRASP-TS.

In order to further evaluate our GRASP-TS and GRASP-TS/PM algorithms, we compare our results with some best performing algorithms in the litera-

Table 2
Computational Results on UBQP Instances

Instance	f_{prev}	GRASP-TS				GRASP-TS/PM			
		f_{best}	f_{avr}	σ	$time$	f_{best}	f_{avr}	σ	$time$
b2500.1	1515944	1515944	1515944	0	12	1515944	1515944	0	12
b2500.2	1471392	1471392	1471138	218	38	1471392	1471257	154	52
b2500.3	1414192	1414192	1414179	58	34	1414192	1414192	0	33
b2500.4	1507701	1507701	1507701	0	11	1507701	1507701	0	10
b2500.5	1491816	1491816	1491816	0	13	1491816	1491816	0	17
b2500.6	1469162	1469162	1469162	0	24	1469162	1469162	0	20
b2500.7	1479040	1479040	1479014	63	34	1479040	1479039	3	60
b2500.8	1484199	1484199	1484198	4	27	1484199	1484199	0	25
b2500.9	1482413	1482413	1482407	6	30	1482413	1482412	4	42
b2500.10	1483355	1483355	1483308	142	31	1483355	1483355	0	56
p3000.1	3931583	3931583	3931573	44	103	3931583	3931583	0	113
p3000.2	5193073	5193073	5193073	0	47	5193073	5193073	0	63
p3000.3	5111533	5111533	5111501	86	103	5111533	5111533	0	153
p3000.4	5761822	5761822	5761822	0	78	5761822	5761822	0	53
p3000.5	5675625	5675625	5675514	162	160	5675625	5675573	180	172
p4000.1	6181830	6181830	6181830	0	128	6181830	6181830	0	141
p4000.2	7801355	7801355	7801098	709	316	7801355	7801332	47	363
p4000.3	7741685	7741685	7741679	19	232	7741685	7741685	0	253
p4000.4	8711822	8711822	8711783	72	357	8711822	8711812	30	321
p4000.5	8908979	8908979	8908376	985	206	8908979	8908643	726	385
p5000.1	8559680	8559680	8558628	554	893	8559680	8558895	422	530
p5000.2	10836019	10836019	10835517	469	553	10836019	10835858	288	760
p5000.3	10489137	10489137	10488369	722	86	10489137	10488780	321	570
p5000.4	12252318	12252318	12250975	635	662	12252318	12251098	641	960
p5000.5	12731803	12731803	12731151	509	478	12731803	12731710	221	804
p6000.1	11384976	11384976	11384218	476	1314	11384976	11384613	205	1415
p6000.2	14333855	14333855	14332637	786	1255	14333855	14333119	843	229
p6000.3	16132915	16132915	16130966	1254	371	16132915	16131166	1224	1350
p7000.1	14478676	14478676	14476478	1128	2798	14478676	14477110	881	2540
p7000.2	18249948	18249948	18247495	1566	2178	18249948	18248499	901	1938
p7000.3	20446407	20446407	20444906	1310	1704	20446407	20445621	720	2809
Average		0*	509.6*	386.4	460.5	0*	316.9*	252.0	524.2

*: The gaps to the previous best result ($f_{prev} - f_{best}$,$f_{prev} - f_{avr}$) are calculated.

ture. Notice that a completely fair comparison is impossible since the reference algorithms are implemented by different authors and run under different conditions. Our comparison here on the UBQP instances as well as that on the MaxCut problem are thus presented only for indicative purposes and should be interpreted with caution. Nevertheless, our experiments provide an indication of the performance of the proposed algorithms relative to the state-of-the-art algorithms.

For this purpose, we restrict our attention to comparisons in terms of quality with six methods that have reported the best results for the most challenging problems. These methods are respectively named ITS [29], MST1 [27], MST2 [27], SA [18], D^2TS [15] and HMA [22]. Moreover, we focus only on the 11 largest and most difficult instances with variables from 5000 to 7000 since the best results for instances with size smaller than 5000 can be easily reached by all these state-of-the art algorithms.

Table 3
Best Results Comparison on Larger UBQP Instances

Instance	f_{prev}	best solution gap (i.e., $f_{prev} - f_{best}$)							
		ITS	MST1	MST2	SA	D^2TS	HMA	GRASP-TS	GRASP-TS/PM
p5000.1	8559680	700	3016	325	1432	325	0	0	0
p5000.2	10836019	0	0	582	582	0	0	0	0
p5000.3	10489137	0	3277	0	354	0	0	0	0
p5000.4	12252318	934	3785	1643	444	0	0	0	0
p5000.5	12731803	0	5150	0	1025	0	0	0	0
p6000.1	11384976	0	3198	0	430	0	0	0	0
p6000.2	14333855	88	10001	0	675	0	0	0	0
p6000.3	16132915	2729	11658	0	0	0	0	0	0
p7000.1	14478676	340	7118	1607	2579	0	0	0	0
p7000.2	18249948	1651	8902	2330	5552	104	0	0	0
p7000.3	20446407	0	17652	0	2264	0	0	0	0
Average		585.6	6705.2	589.7	1394.3	39	0	0	0

Table 3 shows the gap to the best known objective value of our GRASP-TS and GRASP-TS/PM algorithms compared with the reference algorithms. The last row presents the averaged results over the 11 instances. The results of the first 4 reference algorithms are directly extracted from [29], the results of D^2TS are from [15] and the results of HMA come from [22]. Note that the results of all these algorithms are obtained almost under the same time limit.

From Table 3 it is observed that both GRASP-TS and GRASP-TS/PM outperform the 5 reference algorithms (ITS, MST1, MST2, SA and D^2TS) and are also competitive with our HMA algorithm in terms of the quality of the best solution, demonstrating the efficacy of the two GRASP-Tabu Search algorithms in finding the best objective values. In order to further discriminate between GRASP-TS, GRASP-TS/PM and HMA, we compare the average solution gaps (20 independent runs) to the best known objective values over 31 instances. We find that GRASP-TS/PM is slightly better than HMA with a gap of 316.9 against 332.2. Also GRASP-TS is inferior to both GRASP-TS/PM and HMA with a gap of 509.6.

We also apply the Friedman non-parametric statistical test followed by the Post-hoc test to the results in Table 3 to see whether there exists significant performance differences between our proposed algorithms and the reference methods. Firstly, we observe from the Friedman test that there is a significant difference among the compared algorithms (with a p-value of 3.737e-06). Furthermore, the Post-hoc analysis shows that GRASP-TS is significantly better than MST1 and SA (with p-values of 5.330108e-06 and 3.622423e-03, respectively) but is not significantly better than ITS, MST2 and D^2TS (with p-values of 5.347580e-01, 5.347227e-01 and 9.995954e-01, respectively).

Since the best solution values obtained by GRASP-TS, GRASP-TS/PM and HMA are the same, we carry out the above statistical tests with regard to the average solution values. Notice that 31 UBQP instances are considered in this experiment. Firstly, from the the Friedman test, we confirm that there exists a significant performance difference between GRASP-TS, GRASP-TS/PM and HMA (with a p-value of 4.267e-06). Furthermore, the Post-hoc analysis shows that both GRASP-TS/PM and HMA are significantly better than GRASP (with p-values of 4.089688e-06 and 3.296903e-04, respectively). However, we cannot conclude whether GRASP-TS/PM or HMA performs significantly better than the other (with a p-value of 5.999315e-01).

4.4 Computational Results on the MaxCut Instances

In this section, we test GRASP-TS and GRASP-TS/PM on the 54 MaxCut instances and the results of this experiment are summarized in Table 4, using the same statistics as in Table 2. The previous best results are from references [8,10,24,28,33].

From Table 4, we observe that GRASP-TS/PM outperforms GRASP-TS with respect to the best and average objective values. Specifically, GRASP-TS/PM has the best gap relative to the previous best result of 0.78 on average over 54 instances while GRASP-TS has a gap of 5.76. Moreover, GRASP-TS/PM has an average objective gap over 20 runs relative to the previous best known value of 4.50, which is two times smaller than obtained by GRASP-TS with a gap of 9.68. However, GRASP-TS/PM needs slightly more CPU time to reach its best solutions and its objective value variance is slightly larger than GRASP-TS. It is noteworthy that both methods achieve exceedingly high quality outcomes, although GRASP-TS/PM emerges the clear winner. In particular, GRASP-TS/PM improves the previous best known results on 19 instances (in bold), while GRASP-TS improves the previous best known results for 9 instances.

For comparative purposes, Table 5 also includes the results of three state-of-the-art algorithms. These reference methods are Scatter Search [24] (column 3), CirCut heuristic [8] (column 4) and VNSPR [10] (column 5). The last three rows of Table 5 show the summary of the comparison between each algorithm including ours and the previous best known results. The rows *better*, *equal*, *worse* respectively denote the number of instances for which each algorithm gets better, equal and worse results than the previous best known results. The results of these reference algorithms are directly extracted from [24] (results obtained on a personal computer with a 3.2GHz Intel Xenon processor and 2.0 GB of RAM which is comparable to our computer with a Pentium 2.83GHz and 2.0 GB RAM). However, not all the algorithms are run under the same conditions and hence, this comparison should be interpreted with caution. Notice also that while some reference algorithms are MaxCut specific heuristics, our algorithm is designed for the more general UBQP problem.

Table 4
Computational Results on MaxCut Instances

Instance	f_{prev}	GRASP-TS				GRASP-TS/PM			
		f_{best}	f_{avr}	σ	$time$	f_{best}	f_{avr}	σ	$time$
G1	11624	11624	11624.0	0.0	100	11624	11624.0	0.0	47
G2	11620	11620	11619.6	0.7	677	11620	11620.0	0.0	210
G3	11622	11620	11619.9	0.5	854	11620	11620.0	0.0	297
G4	11646	11646	11646.0	0.0	155	11646	11646.0	0.0	49
G5	11631	11631	11631.0	0.0	235	11631	11631.0	0.0	232
G6	2178	2178	2177.4	0.6	453	2178	2177.9	0.2	518
G7	2003	**2006**	2005.9	0.3	304	**2006**	2006.0	0.0	203
G8	2003	**2005**	2004.7	0.5	565	**2005**	2004.9	0.3	596
G9	2048	**2054**	2053.4	0.7	581	**2054**	2053.6	0.7	559
G10	1994	**2000**	1999.3	0.6	845	**2000**	1999.3	0.7	709
G11	564	564	564.0	0.0	18	564	564.0	0.0	10
G12	556	556	555.5	0.9	723	556	556.0	0.0	233
G13	582	582	581.1	1.0	842	582	581.8	0.6	516
G14	3064	3062	3061.6	0.5	812	3063	3062.1	0.4	1465
G15	3050	3040	3037.7	1.0	419	3050	3049.1	0.2	1245
G16	3052	3049	3044.4	1.2	1763	3052	3050.9	0.7	335
G17	3043	3043	3040.6	0.8	1670	**3047**	3045.8	1.1	776
G18	988	**992**	989.3	1.3	977	**992**	992.0	0.0	81
G19	903	**906**	904.4	1.0	490	**906**	906.0	0.2	144
G20	941	941	941.0	0.0	578	941	941.0	0.0	80
G21	931	927	925.7	0.8	484	931	930.6	0.5	667
G22	13359	13346	13336.1	4.9	983	13349	13342.4	3.0	1276
G23	13342	13318	13311.7	3.7	1668	13332	13322.4	4.4	326
G24	13337	13313	13306.0	4.5	643	13324	13317.3	3.7	1592
G25	13326	13315	13306.9	3.8	767	13326	13318.1	3.3	979
G26	13314	13306	13294.8	4.9	1483	13313	13303.3	4.2	1684
G27	3318	3316	3304.2	4.5	256	**3325**	3318.1	3.7	832
G28	3285	3275	3267.8	3.5	82	**3287**	3277.4	3.8	1033
G29	3389	3386	3370.9	7.1	21	**3394**	3384.5	6.0	993
G30	3403	3395	3383.3	4.4	1375	3402	3393.4	4.1	1733
G31	3288	3286	3279.4	3.7	904	**3299**	3287.7	4.2	888
G32	1410	1394	1391.8	1.4	903	1406	1397.3	3.1	1232
G33	1382	1368	1365.6	1.0	1501	1374	1369.1	2.1	506
G34	1384	1376	1371.3	1.7	1724	1376	1372.5	2.2	1315
G35	7684	7653	7648.6	2.6	1124	7661	7657.4	2.7	1403
G36	7677	7646	7641.1	2.4	543	7660	7652.1	5.1	1292
G37	7689	7664	7657.1	2.4	983	7670	7662.0	4.1	1847
G38	7681	7653	7644.3	4.0	667	7670	7659.8	4.8	1296
G39	2395	2388	2381.9	2.5	911	**2397**	2387.1	5.0	742
G40	2387	2378	2359.6	5.8	134	**2392**	2384.3	5.8	1206
G41	2398	2367	2355.3	6.9	612	2398	2383.7	8.2	1490
G42	2469	2453	2447.5	2.9	1300	**2474**	2461.7	5.6	1438
G43	6660	6660	6658.3	1.0	969	6660	6659.4	0.7	931
G44	6650	6649	6647.1	1.1	929	6649	6647.7	0.8	917
G45	6654	6654	6652.5	0.8	1244	6654	6652.6	0.7	1791
G46	6645	**6648**	6645.4	1.4	702	**6649**	6646.0	1.7	405
G47	6656	6656	6654.5	1.0	1071	6656	6655.4	0.7	725
G48	6000	6000	6000.0	0.0	13	6000	6000.0	0.0	4
G49	6000	6000	6000.0	0.0	27	6000	6000.0	0.0	6
G50	5880	5880	5880.0	0.0	80	5880	5880.0	0.0	14
G51	3846	3843	3839.3	1.9	628	**3847**	3843.8	1.5	701
G52	3849	3844	3840.6	1.5	1274	**3850**	3846.8	1.9	1228
G53	3846	**3847**	3844.3	1.3	1317	**3848**	3845.8	1.0	1419
G54	3846	**3848**	3845.6	1.2	1231	**3850**	3847.8	1.9	1215
Average		5.76*	9.68*	1.89	770.6	0.78*	4.50*	1.96	804.3

*: The gaps to the previous best result ($f_{prev} - f_{best}$, $f_{prev} - f_{avr}$) are calculated.

Table 5
Best Results Comparison on MaxCut Instances

Instance	f_{prev}	best solution value				
		SS	CirCut	VNSPR	GRASP-TS	GRASP-TS/PM
G1	11624	11624	11624	11621	11624	11624
G2	11620	11620	11617	11615	11620	11620
G3	11622	11622	11622	11622	11620	11620
G4	11646	11646	11641	11600	11646	11646
G5	11631	11631	11627	11598	11631	11631
G6	2178	2165	2178	2102	2178	2178
G7	2003	1982	2003	1906	**2006**	**2006**
G8	2003	1986	2003	1908	**2005**	**2005**
G9	2048	2040	2048	1998	**2054**	**2054**
G10	2000	1993	1994	1910	**2000**	**2000**
G11	564	562	560	564	564	564
G12	556	552	552	556	556	556
G13	582	578	574	580	582	582
G14	3064	3060	3058	3055	3062	3063
G15	3050	3049	3049	3043	3040	3050
G16	3052	3045	3045	3043	3049	3052
G17	3043	3043	3037	3030	3043	**3047**
G18	988	988	978	916	**992**	**992**
G19	903	903	888	836	**906**	**906**
G20	941	941	941	900	941	941
G21	931	930	931	902	931	931
G22	13359	13346	13346	13295	13346	13349
G23	13342	13317	13317	13290	13318	13332
G24	13337	13303	1314	13276	13313	13324
G25	13326	13320	13326	12298	13315	13326
G26	13314	13294	13314	12290	13306	13313
G27	3318	3318	3306	3296	3316	**3325**
G28	3285	3285	3260	3220	3275	**3287**
G29	3389	3389	3376	3303	3389	**3394**
G30	3403	3403	3385	3320	3395	3402
G31	3288	3288	3285	3202	3286	**3299**
G32	1410	1398	1390	1396	1394	1406
G33	1382	1362	1360	1376	1368	1374
G34	1384	1364	1368	1372	1376	1376
G35	7684	7668	7670	7635	7653	7661
G36	7677	7660	7660	7632	7646	7660
G37	7689	7664	7666	7643	7664	7670
G38	7681	7681	7646	7602	7653	7670
G39	2395	2393	2395	2303	2388	**2397**
G40	2387	2374	2387	2302	2378	**2392**
G41	2398	2386	2398	2298	2367	2398
G42	2469	2457	2469	2390	2453	**2474**
G43	6660	6656	6656	6659	6660	6660
G44	6650	6648	6643	6642	6649	6649
G45	6654	6642	6652	6646	6654	6654
G46	6645	6634	6645	6630	**6648**	**6649**
G47	6656	6649	6656	6640	6656	6656
G48	6000	6000	6000	6000	6000	6000
G49	6000	6000	6000	6000	6000	6000
G50	5880	5880	5880	5880	5880	5880
G51	3846	3846	3837	3808	3843	**3847**
G52	3849	3849	3833	3816	3844	**3850**
G53	3846	3846	3842	3802	**3847**	**3848**
G54	3846	3846	3842	3820	**3848**	**3850**
Better	—	0	0	0	9	19
Matched	—	22	20	6	18	20
Worse	—	32	34	48	27	15

Table 5 discloses that GRASP-TS/PM and GRASP-TS can find new best results on 19 and 9 instances, respectively among the 54 instances and both match the previous best known results on 20 and 18 instances. For the tested instances, both GRASP-TS/PM and GRASP-TS perform better than the reference algorithms. In particular, GRASP-TS/PM (GRASP-TS respect.) fails to reach the best known results for 15 (27 respect.) instances while the reference algorithms SS, CirCut and VNSPR fail on 32, 34 and 48 instances, respectively. The computing times (in seconds) to reach the best solution of GRASP-TS (770) and GRASP-TS/PM (804) are larger than SS (621) and CirCut (616) but much smaller than VNSPR (64505).

As for Table 3, we apply the Friedman test and the Post-hoc test to the results in Table 5 to see whether there are significant performance differences between the proposed methods and other competitors on the 54 MaxCut instances. Firstly, we discover from the Friedman test that SS, CirCut, VNSPR, GRASP-TS and GRASP-TS/PM demonstrate significant differences (with a p-value of 2.2e-16). Secondly, when comparing GRASP-TS with SS, CirCut and VNSPR, the Post-hoc analysis indicates that GRASP-TS is significantly better than VNSPR (with a p-value of 3.788002e-10) but is not significantly better than SS and CirCut (with p-values of 4.534268e-01 and 9.358923e-02, respectively). Thirdly, when comparing GRASP-TS/PM with SS, CirCut and VNSPR, the Post-hoc analysis indicates that GRASP-TS/PM is significantly better than SS, CirCut and VNSPR (with p-values of 4.059707e-06, 2.433377e-08, 0.000000e+00, respectively). Finally, we observe that GRASP-TS/PM is significantly better than GRASP-TS (with a p-value of 6.795472e-03).

In sum, the computational results on the 85 random and structured instances demonstrate the efficacy of our proposed GRASP-Tabu Search algorithms for solving the UBQP problems, with GRASP-TS/PM emerging as superior to the other methods studied in our comparative tests.

5 Conclusion

In this paper, we studied a simple and a population-based GRASP-Tabu Search algorithm for solving the UBQP problem. Both algorithms are based on a dedicated randomized greedy construction heuristic, enhanced by reference to the ideas of "strongly determined variables" and "elite solution recovery" of probabilistic Tabu Search, and using a tabu search local optimization procedure. Additionally, the algorithm with population management (GRASP-TS/PM) integrates a population management strategy for maintaining a pool of diversified elite solutions.

Tested on three sets of 85 well-known random and structured benchmark instances, we have shown that both GRASP-Tabu Search algorithms obtain highly competitive results in comparison with the previous best known results from the literature. In particular, for the 54 structured instances derived from MaxCut, GRASP-TS/PM can improve the best known objective values for 19 instances whose optimum solution values are still unknown. In future work, we look forward to exploiting other forms of population-based search strategies like Path Relinking and more advanced tabu search mechanisms to provide further gains along these lines.

Acknowledgment

We would like to thank the anonymous referees for their helpful comments and suggestions. The work is partially supported by the regional RaDaPop (2009-2012) and LigeRo projects (2009-2012).

References

[1] Alidaee B, Kochenberger GA, Ahmadian A (1994) 0-1 quadratic programming approach for optimum solutions of two scheduling problems. International Journal of Systems Science 25(2):401–408

[2] Amini M, Alidaee B, Kochenberger GA (1999) A scatter search approach to unconstrained quadratic binary programs, New Ideas in Optimization, London: McGraw-Hill, pp 317–330

[3] Beasley JE (1996) Obtaining test problems via Internet. Journal of Global Optimization 8(4):429–433

[4] Beasley JE (1998) Heuristic algorithms for the unconstrained binary quadratic programming problem. Working Paper, The Management School, Imperial College, London, England

[5] Borgulya I (2005) An evolutionary algorithm for the unconstrained binary quadratic problems. Advances in Soft Computing 33:3–16

[6] Boros E, Hammer PL, Sun R, Tavares G (2008) A max-flow approach to improved lower bounds for quadratic unconstrained binary optimization. Discrete Optimization 5(2):501–529

[7] Boros E, Hammer PL, Tavares G (2007) Local search heuristics for Quadratic Unconstrained Binary Optimization (QUBO). Journal of Heuristics 13:99–132

[8] Burer S, Monteiro RDC, Zhang Y (2002) Rank-two relaxation heuristics for max-cut and other binary quadratic programs. SIAM Journal on Optimization 12:503–521

[9] Chardaire P, Sutter A (1995) A decomposition method for quadratic zero-one programming. Management Science 41(4):704-712

[10] Festa P, Pardalos PM, Resende MGC, Ribeiro CC (2002) Randomized heuristics for the Max-Cut problem. Optimization Methods and Software 17(6):1033–1058

[11] Glover F (1977) Heuristics for integer programming using surrogate constraints. Decision Sciences 8(1):156–166

[12] Glover F (1989) Tabu Search - Part I. ORSA Journal on Computing 1(3):190–206

[13] Glover F, Laguna M (1997) Tabu search. Kluwer Academic Publishers, Boston

[14] Glover F, Kochenberger GA, Alidaee B (1998) Adaptive memory tabu search for binary quadratic programs. Management Science 44:336–345

[15] Glover F, Lü Z, Hao JK (2010) Diversification-driven tabu search for unconstrained binary quadratic problems. 4OR: A Quarterly Journal of Operations Research 8:239–253

[16] Harary F (1953) On the notion of balance of a signed graph. Michigan Mathematical Journal 2(2):143–146

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

[18] Katayama K, Narihisa H (2001) Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem. European Journal of Operational Research 134(1):103–119

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

[20] Krarup J, Pruzan PM (1978) Computer-aided layout design. Mathematical Programming Studies 9:75–94

[21] Lodi A, Allemand K, Liebling TM (1999) An evolutionary heuristic for quadratic 0-1 programming. European Journal of Operational Research 119(3):662–670

[22] Lü Z, Glover F, Hao JK (2010) A hybrid metaheuristic approach to solving the UBQP problem. European Journal of Operational Research 207(3):1254–1262

[23] McBride RD, Yormark JS (1980) An implicit enumeration algorithm for quadratic integer programming. Management Science 26:282–296

[24] Marti R, Duarte A, Laguna M (2009) Advanced scatter search for the max-cut problem. INFORMS Journal on Computing 21(1):26–38

[25] Merz P, Freisleben B (1999) Genetic algorithms for binary quadratic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO99), Morgan Kaufmann, pp 417–424

[26] Merz P, Katayama K (2004) Memetic algorithms for the unconstrained binary quadratic programming problem. Biosystems 78:99–118

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

[28] Palubeckis G (2004) Application of multistart tabu search to the MaxCut problem. Information Technology and Control 2(31):29–35

[29] Palubeckis G (2006) Iterated tabu search for the unconstrained binary quadratic optimization problem. Informatica 17:279–296

[30] Pardalos P, Rodgers GP (1990) Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing 45:131–144

[31] Resende M, Ribeiro C (2003) Greedy randomized adaptive search procedures. Handbook of Metaheuristics 57:219–249

[32] Resende M, Ribeiro C (2005) Grasp and path-relinkig: recent advances and applications. Metaheuristics: progress as real problem solvers

[33] Shylo VP, Shylo OV (2010) Solving the maxcut problem by the global equilibrium search. Cybernetics and Systems Analysis 46(5):744–754

[34] Wang Y, Lü Z, Glover F, Hao JK (2011) Backbone guided Tabu Search for solving the UBQP problem. Journal of Heuristics (DOI: 10.1007/s10732-011-9164-4)

[35] Wang Y, Lü Z, Glover F, Hao JK (2011) Effective variable fixing and scoring strategies for binary quadratic programming. P. Merz, J.K. Hao (Eds): EvoCOP 2011, LNCS 6622: 72–83, Springer

[36] Xu JF, Chiu SY, Glover F (1996) Probabilistic tabu search for telecommunications network design. Combinational Optimization: Theory and Practice 1(1):69–94

[37] Xu JF, Chiu SY, Glover F (1998) Fine-tuning a tabu search algorithm with statistical tests. Internatioanl Transactions in Operational Research 5:233–244