Computers & Industrial Engineering 111 (2017) 164–175

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Simple and fast novel diversification approach for the UBQP based on sequential improvement local search

Bahram Alidaee [a,*], Hugh Sloan [a], Haibo Wang [b]

[a] Marketing Department, School of Business Administration, The University of Mississippi, P.O. Box 1848, University, MS 38677-1848, United States
[b] College of Business Administration, Texas A & M International University, Laredo, TX 78041, United States

ARTICLE INFO

Article history:
Received 9 September 2015
Received in revised form 10 July 2017
Accepted 11 July 2017
Available online 15 July 2017

Keywords:
Unconstrained binary quadratic program
Combinatorial optimization
Sequential local search
Maximum cut problem

ABSTRACT

The unconstrained binary quadratic program (UBQP) is a challenging NP-hard problem. Due to its vast applicability and the theoretical interests it has grown in importance in the recent years. Various heuristics have been proposed as solution procedures. Most of the heuristics are based on local improvement procedures. To be able to reach optimal or near optimal solutions, researchers have implemented multi-start and diversification strategies to explore a larger solution space. Diversification strategies in the literature concentrate on some manipulation of solutions (variables). In this paper, relationship between starting solution, the sequence of implementing local search, and the locally optimal solution x^* is explored. A novel diversification approach based on the sequence to implement the local improvement is proposed. We implemented four versions of our diversification procedures within a simple tabu search and tested on several benchmark problems available on the Internet. Our extensive computational results show that the procedures can reach the best known solutions with high frequencies within very short CPU time. For 123 out of 125 of these problems the procedures reached the best known solutions quickly. For 44 of the 84 Max-Cut benchmark problems the procedures improved upon the available solutions in reasonably short CPU time.

1. Introduction

The unconstrained binary quadratic programming (UBQP) problem is defined by

$$Max\ x^tQx,\ s.t.,\ x_i \in \{0,1\},\ \text{for } i = 1,\dots,n. \tag{1}$$

In (1), Q is an $n \times n$ symmetric matrix, x is a binary vector of n variables, and x^t is the transpose of x. The UBQP, (also known in the literature as the *xQx model*, see for example, Kochenberger & Glover, 2013; Lewis, Alidaee, & Kochenberger, 2005; Lewis, Alidaee, Glover, & Kochenberger, 2009) has served as a unifying model for many combinatorial problems. It plays an important theoretical and practical role in combinatorial optimization, (Hasan, Alkhamis, & Ali, 2000). Many industrial problems including the well-known quadratic assignment problem can be formulated as UBQP, and solved efficiently (Aksan, Dokeroglu, & Cosar, 2017; Bayram & Sahin, 2016; Branda, Novotney, & Olstad, 2016; Das, 2017; Hasan et al., 2000; Kumar, Rosenberger, & Iqbal, 2016; Lewis et al., 2005, 2009). Furthermore, many applications of the problem from other areas of optimization has been reported, e.g., Hopfield network, graph theory, set partitioning, layout analysis, coloring problem, circuit layout design, gate assignment, land use planning, cell manufacturing system design, (Ahmadi, Pishvaee, & Akbari Jokar, 2017; Bayram & Sahin, 2016; Branda et al., 2016; Das, 2017; Kumar et al., 2016; Singh & Sharma, 2006), to name a few. In this paper we address the problem (1) interchangeably using UBQP and *xQx* models. In the last decade various optimal and heuristic algorithms have been proposed for the *xQx* model. Refer to two recent survey papers by Kochenberger, Glover, Alidaee, and Rego (2004), and Kochenberger et al. (2014) for various applications and solution procedures.

Due to the challenging nature of the UBQP, researchers have developed heuristics to solve the model. Although many different heuristics and meta-heuristics have been applied to the model, however, most of heuristics are some variants of tabu search (TS) including scatter search (SS) procedures. TS/SS has proven to be successful on many applications of UBQP, (see Kochenberger et al., 2014). Several very fast one-pass procedures also have been proposed to solve the model, (Boros, Hammer, & Sun, 1989; Glover, Alidaee, Rego, & Kochenberger, 2002; Hanafi, Rebai, & Vasquez, 2013).

* Corresponding author.
E-mail addresses: balidaee@bus.olemiss.edu (B. Alidaee), hsloan@bus.olemiss.edu (H. Sloan), hwang@tamiu.edu (H. Wang).

http://dx.doi.org/10.1016/j.cie.2017.07.012
0360-8352/

In order to reach a global or near global solution in combinatorial optimization, researchers apply multistart and diversification strategies to explore larger solution space, (Bui & Moon, 1996; Chou, Chien, & Gen, 2014; Etscheid & Roglin, 2014; Glover, Lü, & Hao, 2010; GroSelj & Malluhi, 1995; Hagen & Kahng, 1997; Hwang, Alidaee, & Johnson, 1999; James, Rego, & Glover, 2009; Li & Zhang, 2012; Lin & Zhu, 2014; Luo, Cheang, Lim, & Zhu, 2013; Nguyen, Zhang, Johnston, & Tan, 2015; Pan, Wang, Sang, Li, & Liu, 2013; Qureshi, Mirza, Rajpoot, & Arif, 2011; Ren, Jiang, Xuan, Hu, & Luo, 2014; Shi & Pan, 2005; Sun, Zhang, & Yao, 2014; Xudong, Guangzheng, & Shiyou, 2002; Yuan & Xu, 2015; Zhou, Lai, & Li, 2015). Diversification strategy is also one component of metaheuristic search processes, (Kochenberger & Glover, 2013; Marti, Durante, & Laguna, 2009; Shylo & Shylo, 2014; Wang, Lu, Glover, & Hao, 2012). In particular, diversification strategies in TS/SS are well documented to be effective in solving combinatorial optimization problems, e.g., the *xQx* model, (Glover et al., 2010; Katayama, Tani, & Narihisa, 2000; Lü, Glover, & Hao, 2010; Marti et al., 2009). To the best of our knowledge all diversification strategies in the literature concentrate on clever manipulations of solutions (variables) to explore a larger solution space, (Etscheid & Roglin, 2014; GroSelj & Malluhi, 1995; James et al., 2009; Katayama et al., 2000; Marti et al., 2009; Merz & Freisleben, 2002; Palubeckis, 2004; Palubeckis, 2006; Ren et al., 2014; Shi & Pan, 2005; Wang & Lim, 2008). Note that multistart strategy is also a way of implementing some diversification strategy. However, in the following we will show that besides manipulation of variables the sequence of steps to implement the local improvement can significantly explore a larger solution space. A careful implementation of a combination of multistart and sequence of local improvement process can create many diverse solutions. Specifically, in this paper we carefully implement a combination of the following two procedures within a simple tabu search.

1. Borrowing from sequencing problems, e.g., TSP, we adopt a limited versions of *2-Opt, 3-Opt, 4-Opt* and combination of the three heuristics (in the paper this combination is indicated as *ALL*) for the *xQx* mode.
2. We implement a multistart solution strategy as, all 0's, all 1's, and randomly chosen 0–1 for x_i.

We also point out that, in our practice we have noticed that randomly changing the components of solution x as the search progresses can provide better solutions to the *xQx* model. This random change of solution also creates new starting solution as the search progresses. Thus, we also included a random change of variables in the solution as part of our strategy. Such random change of variables is adopted from strategic oscillation that have shown to be very effective in binary optimization, in particular UBQP, Glover, Kochenberger, and Alidaee (1998).

Careful implementations of local optimal procedures in combinatorial optimization have proven to be very powerful solution approaches. Regarding sequencing problems, such as TSP, such implementations mostly have roots in *n-Opt* strategies. Two characteristics of *n-Opt* strategies are *(i)* easily implementable and *(ii)* ability to expand the search on the solution space and thus creates diverse solutions. Local search solution procedures for *xQx* models mostly have roots in the *r-flip* heuristics where one or several x_i's are changed to $1-x_i$ (refer to Alidaee, Kochenberger, and Wang (2010), for theoretical developments of *r-flip* implementations on Pseudo-Boolean optimization). The *n-Opt* strategies that over the years have been applied to sequencing problems is fundamentally different from *r-flip* strategies. The main purpose of the present paper is to appropriately adopt *n-Opt* strategies borrowed from sequencing problems and combine with *1-flip* strategy for *xQx* models. To illustrate, a naïve implementation of *2-Opt* strategy for *xQx* model may be explained as follows. Consider first a sequencing problem, for example, an n-job sequencing problem where d_{ij} is a penalty (e.g., set-up cost) if job j is scheduled after job i. Also, assume that d_{ij} may be different from d_{ji}. Different sequences of jobs provide different total penalties. There are potentially $n!$ different sequences, and thus values for the total penalty. To implement a *2-Opt* heuristic strategy we may randomly choose a sequence of the jobs then randomly choose two jobs, e.g., job k and job j, where job j is sequenced somewhere after job k. Then create a new sequence by reversing all jobs from k to j. The search continues until a stopping criteria is reached. To adopt this strategy for *xQx* model we may start with a random sequence of the n variables, then implement *1-flip* local search one after another according to this sequence. Then randomly change the sequence using a *2-Opt* and again implement the *1-flip* local search strategy according to the new sequence. Continue the process until we have reached a local optimal solution.

In the next section through an example we demonstrate that different orders (sequences) of updating the local improvement steps in the *xQx* model combined with a multistart strategy can significantly affect the final solutions. Many researchers have applied variations of *n-Opt* strategies to sequencing problems. Due to enormous number of possible options to implement an *n-Opt* strategy, researchers concentrate on some limited versions of *2-Opt*, *3-Opt* and *4-Opt* strategies. In this paper, we adopt these strategies for *xQx* model. We implement 5 possible ordering choices for *2-Opt*, Fig. 2, 24 possible ordering choices for *3-Opt*, Fig. 3, and 8 possible ordering choices for the so called double bridge version of *4-Opt* strategies, Fig. 4. Depending on how it is implemented each of the 37 possible strategies creates enormous possible sequencing update orders and thus expand the search to bigger solution space. In the next section we formally introduce the diversification procedures.

2. A diversification processes based on sequential approach for UBQP

We first explain condition of local optimality in *xQx* problem when *1-flip* local search is applied. Given a *Max xQx* problem and a binary solution x, where q_{ij} is the ij-th element of matrix Q, define

$$\Delta_i(x) = q_{ii} + \sum_{j \neq i} 2q_{ij}x_j, \text{ for } i = 1, \ldots, n. \tag{2}$$

It is well known that a solution x is locally optimal if and only if the following condition is satisfied

$$x_i = 1, \text{ for all } \Delta_i(x) > 0, \text{ and } x_i = 0, \text{ for all } \Delta_i(x) \leqslant 0. \tag{3}$$

Given a solution x if condition (3) for a variable x_i is not satisfied the *1-flip* local search process changes x_i to $1-x_i$ one at a time in some order and the amount of improvement to the objective function for each change is equal to $\Delta_i(x)$. Furthermore update for the vector $\Delta(1-x) = \Delta(x_1, \ldots, 1-x_i, \ldots, x_n)$ is calculated as follows.

$$\begin{aligned} \Delta_j(1-x) &= \Delta_j(x) + 2(1-2x_i)q_{ji}, \text{ for } j < i, \\ \Delta_j(1-x) &= \Delta_j(x) + 2(1-2x_i)q_{ij}, \text{ for } j > i. \end{aligned} \tag{4}$$

Using Example 1, we first demonstrate that a combination of different multistart and sequence of updating *1-flip* improvement steps can provide different solutions.

Example 1. Consider a *Max xQx* example from Pardalos and Rodgers (1990), (see Fig. 1 in the Appendix for matrix Q). Objective value of an optimal solution is 1684. Consider two starting solutions, $x_i = 1$ for all i, and $x_i = 0$ for all i. Given a starting sequence of variables, apply four *1*-flip improvement as follows: (1) always implement *1*-flip improvement from left to right, (2)

-30	25	-34	-33	47	-50	-10	-2	40	47	0	43	38	49	-18	38	26	-4	24	21
25	-7	25	14	30	28	-27	39	2	15	-21	-14	-25	45	-21	45	-13	3	11	0
-34	25	-26	-23	13	41	-25	-45	22	-47	27	26	-25	27	-34	-26	10	48	32	-11
-33	14	-23	-35	4	-41	-42	-23	-17	-11	-31	-17	-28	-43	9	22	-42	1	-36	30
47	30	13	4	31	-43	24	45	-25	-35	-25	4	-21	-40	5	46	32	-18	2	7
-50	28	41	-41	-43	19	-33	-49	25	30	-47	44	38	50	-23	-36	-18	-33	-23	1
-10	-27	-25	-42	24	-33	-4	-30	-10	43	18	47	-47	-26	-17	-18	6	-14	32	-35
-2	39	-45	-23	45	-49	-30	-17	-37	-1	-16	25	-5	23	26	-34	-32	42	-23	47
40	2	22	-17	-25	25	-10	-37	-30	-7	27	3	-12	13	-40	-28	11	41	-32	-8
47	15	-47	-11	-35	30	43	-1	-7	25	13	35	1	17	-34	50	35	-24	41	0
0	-21	27	-31	-25	-47	18	-16	27	13	32	-11	-29	-28	31	43	-4	-45	-34	26
43	-14	26	-17	4	44	47	25	3	35	-11	-47	36	-21	26	-20	1	5	29	14
38	-25	-25	-28	-21	38	-47	-5	-12	1	-29	36	40	-19	24	-8	38	46	-2	-48
49	45	27	-43	-40	50	-26	23	13	17	-28	-21	-19	-16	36	38	33	-35	-11	32
-18	-21	-34	9	5	-23	-17	26	-40	-34	31	26	24	36	-31	-31	44	-33	0	-17
38	45	-26	22	46	-36	-18	-34	-28	50	43	-20	-8	38	-31	37	32	-36	-6	5
26	-13	10	-42	32	-18	6	-32	11	35	-4	1	38	33	44	32	13	-3	-9	-13
-4	3	48	1	-18	-33	-14	42	41	-24	-45	5	46	-35	-33	-36	-3	-3	37	-40
24	11	32	-36	2	-23	32	-23	-32	41	-34	29	-2	-11	0	-6	-9	37	20	-27
21	0	-11	30	7	1	-35	47	-8	0	26	14	-48	32	-17	5	-13	-40	-27	22

Fig. 1. Matrix Q for the example used in Table 1.

always implement *1*-flip improvement from right to left, (3) implement *1*-flip improvement using always the most improving element i, and (4) implement *1*-flip improvement using always the least improving element i. Thus, there are total of 8 cases to reach a local optimal solution. Results are shown in Table 1. As the results show both choices of starting solution and improvement sequences can significantly affect the final solutions. Proposition 1 below provides some relationship between starting solution, a sequence of updating *1*-flip local search and the local optimal solution reached.

Proposition 1. Given a local optimal solution x^*, there always exits a starting solution and a sequence where implementation of 1-flip rule reaches x^* in $O(n)$ time.

Proof. For the given locally optimal solution x^* of course condition (3) is satisfied. Now, in some order change (flip) one at a time a variable x_i to $1 - x_i$ if condition (3) is satisfied while after each change update vector Δ using (4). Continue this process at most n times or until there is no variable that condition (3) is satisfied. Let the final solution be x'. Keep track of the sequence, $\pi = (\pi_1, \ldots, \pi_n)$, where change of variables occurred. It should be clear now that, if we now start from x' and flip each variable one at a time using reverse order of π we reach in $O(n)$ time the locally optimal solution x^*.

Note that Proposition 1 does not provide a local optimal solution x^*. Finding a locally optimal solution for xQx in fact can take an exponential number of steps to terminate and the problem of computing a local optimum is PLS-complete, (Etscheid & Roglin, 2014). The proposition however provides a relationship between starting solution, the sequence of implementing *1*-flip local search, and the locally optimal solution x^* that can be reached in $O(n)$ time. It also should be clear from the proof that there are many possible starting solutions and at least a sequence for each case that *1*-flip rule can reach the local optimal solution x^* in $O(n)$ time. The significance of the proposition is the fact that multistart strategy combined with sequence of implementation of *1*-flip rule provides an opportunity to explore a large solution space, that can be considered a diversification generator, for reaching 'quickly' to locally optimal solutions and hopefully to a global or near global solution. Two questions should be considered open that need to be answered in the future research. Given a local optimal solution x^*, and a starting solution x', is there a sequence π that applying *1*-flip rule can reach x^* in $O(n)$ time? Given a local optimal solution x^*, and a sequence π, is there a starting solution x' that applying *1*-flip rule can reach x^* in $O(n)$ time? Although as was explained finding a local optimum solution is difficult however answering these two questions can provide opportunity to create clever procedures to explore larger solution space in shorter time and reaching local and possibly global optimum solutions in faster time. In the following we explain how we applied 2, 3, and 4-*Opt* strategies within a simple tabu search metaheuristic. We first explain different sequencing rules that we adopted from TSP for the xQx model. Then we explain a pseudo code implementation of the procedures.□

2.1. Sequencing rules used with 1-flip improvement procedure

As we explained, to find a local optimal solution using *1-flip* rule we start with a binary solution x and in some *fashion* one at a time change (flip) x_i to $1 - x_i$ if the local optimality condition (3) is not satisfied. The process continues until there is no element i that can

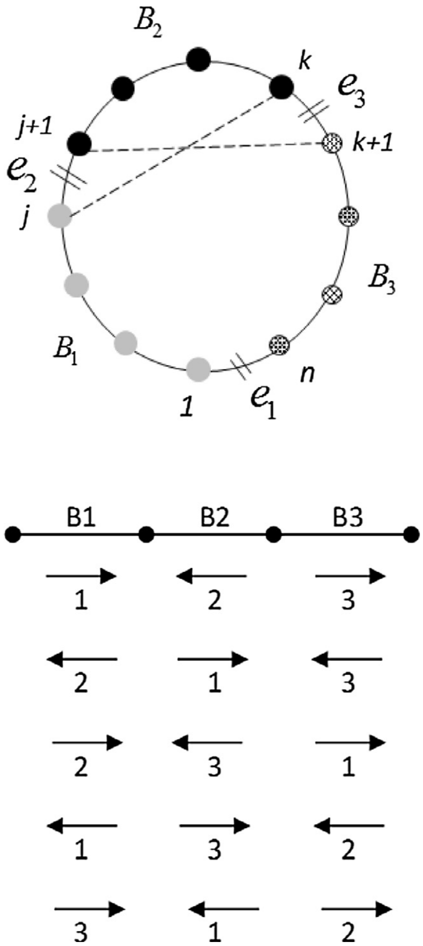

Fig. 2. Implementation of a limited *2_Opt* strategy.

be changed. The order (sequence) to implement the *1-flip* rule is the basis for our procedures. Many researchers use in some form the next most improving element to change, see for example Glover et al. (1998). Example 1 clearly showed that the starting solution as well as the order (sequence) to implement the *flip* rule can make significant difference in reaching final solutions. Implementing improvement in different orders can create different solutions at each step and thus is considered a diversification solution generation for the problem. Over the years researchers have applied ordering processes to generate diversified solutions in sequencing such as traveling salesman problems. The *n-Opt* strategies have been very powerful solution procedures especially when implemented within metaheuristics to solve sequencing and scheduling problems. In the present paper we are adopting such strategies for the UBQP models when the choice of improvement is *1-flip* rule.

At any stage of a process let $x = (x_1, \ldots, x_n)$ be a solution and $\pi = (\pi_1, \ldots, \pi_n)$ a sequence of the *1*,…,*n* numbers where π_i is the position of the *i*-th variable. Assume that we are in the process of implementing a *2-Opt* strategy, and we have just applied the *1-flip* rule one at a time according to the sequence π. Next, we want again to apply the *1-flip* rule. We first randomly choose two numbers $1 \leqslant j < k < n$. In sequence π we now create 3 blocks of variables $B_1 = (\pi_1, \ldots, \pi_j)$, $B_2 = (\pi_{j+1}, \ldots, \pi_k)$, and $B_3 = (\pi_{k+1}, \ldots, \pi_n)$, see Fig. 2. We randomly implement one of the 5 choice of sequences as shown in Fig. 2. For example if choice 1 is chosen we start to implement *1-flip* rule clockwise (*right arrow*) on variables in block 1, then counterclockwise (*left arrow*) on variables in block 2, and finally clockwise (*right arrow*) on variables in block 3. Similarly, if choice 2 is to be implemented we start to implement *1-flip* rule clockwise (*right arrow*) on variables in block 2, then counterclockwise (*left arrow*) on variables in block 1, and finally counterclockwise (*left arrow*) on variables in block 3. These five *2-Opt* choices create many possible sequences as the search continues. Thus as the search continuous implementing *1-flip* rule according to each sequence creates many possible diverse solutions *x*. In the similar fashion we can apply *3-Opt* or *4-Opt* strategies. In *3-Opt* we are limiting the process to 24 cases, Fig. 3, and in *4-Opt* to only 8 possible cases, Fig. 4, which is a limited version of the so called double-bridge *4-Opt* moves. We also applied the process to randomly choosing one of the 37 possible moves, which we called *ALL*.

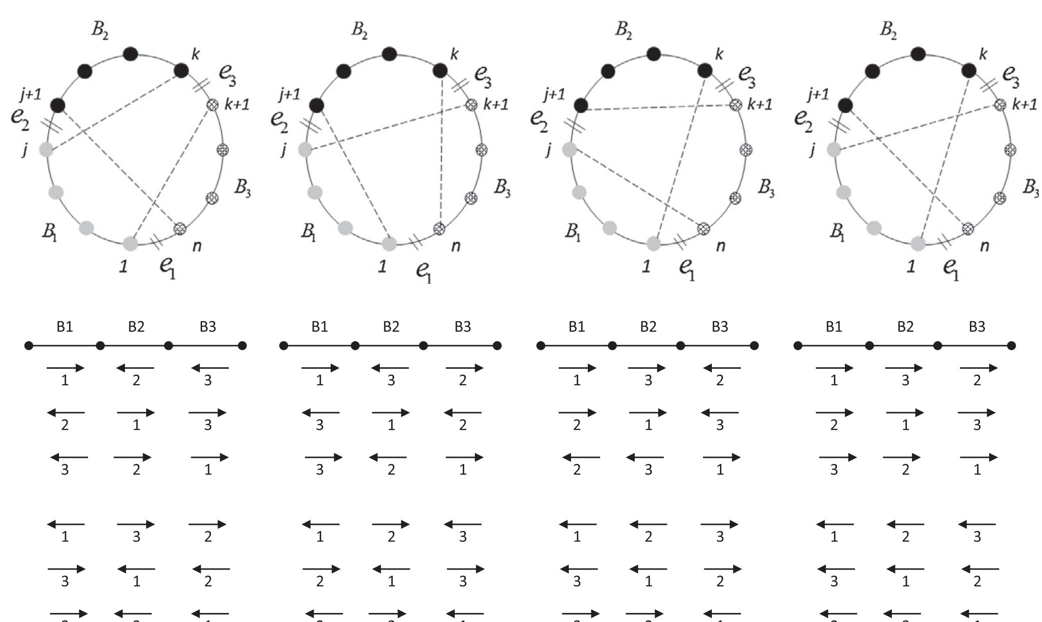

Fig. 3. Implementation of a limited *3_Opt* strategy.

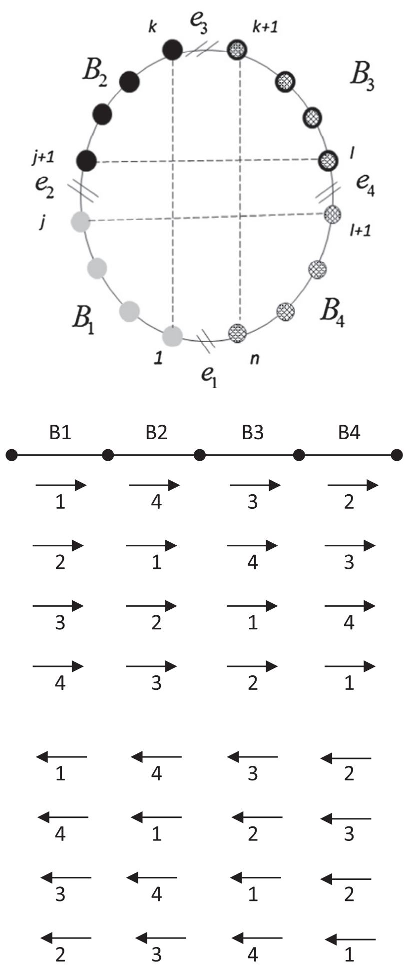

Fig. 4. Implementation of a limited double-bridge *4_Opt* strategy.

Table 1
Result of applying local search with different starting solutions and different improvement sequences.

Starting solution	Implementing sequence	Objective value
X=1	Left-to-right	**1684**
X=1	Right-to-left	1530
X=0	Left-to-right	1616
X=0	Right-to-left	**1684**
X=1	Most improving first	1646
X=1	Least improving first	**1684**
X=0	Most improving first	1601
X=0	Least improving first	**1684**

2.2. A simple tabu search with sequencing rule and multistart strategy

To present a full pseudo code for our implementation of the procedures we first present some definitions:

n, number of variables,
x, a binary starting solution with n variables,
x^*, best solution found so far by an algorithm,
$Z = xQx$, value of the objective function for the variable x,
$Z^* = x^*Qx^*$, value of the objective function for the best solution found,
$J_{next}(x)$, set of variables where conditions of local optimality (3) are not satisfied for a given x,

$$J_{next}(x) = \{i : (\Delta_i(x) < 0 \text{ and } x_i = 1) \text{ or } (\Delta_i(x) > 0 \text{ and } x_i = 0)\},$$

$p1$, $p2$, two integer constants where $p1 < p2 \leqslant n$, their values depend on the problem,
Tabu_ten, a constant integer number as tabu tenure (maximum value a variable can remain tabu),
Tabu(i), for $i = 1,\ldots,n$, a vector representing tabu status of variables,
π, a sequence of $1,\ldots,n$.

In Fig. 5, we present a pseudo code of the simple tabu search that we applied to the 4 procedures. In the pseudo code the "*N_Opt*" calls for a change on sequence π. This is done after each complete *1-flip* iteration, $1,\ldots,n$. If the algorithm is in the process of implementing *2-Opt*, *3-Opt*, or *4-Opt* strategies, we randomly choose one of the 5 choices for *2-Opt*, one of the 24 choices for *3-Opt* and one of the 8 choices for *4-Opt* moves, respectively, to change the sequence (Figs. 2–4). However, if the algorithm is in the process of implementing *ALL* we randomly choose one of the 37 moves to change the sequence. In each case the output is a new sequence π that provides an order to implement the *1-flip* improvement process.

The procedure also randomly chooses a set of variables, x_i, and change (*flip*) to $1 - x_i$. This is done after the local optimality condition is reached by calling "*random_variable_change(.)*". This is done according to the following rule. When a local optimality is reached a random number, p, is generated in the interval $p \in [1, K]$ where K is changing in the interval *[p1, p2]* then if the last variable was changed from 0 to 1 (i.e., *Last_x_changed = 1) then* we randomly change p variables with values $x_i = 0$ to 1, and if the last variable was changed from 1 to 0 (i.e., *Last_x_changed = 0) then* we randomly change p variables with values $x_i = 1$ to 0. The idea is closely related to the strategic oscillation used in many tabu search settings, (see for example, Glover et al. (1998), for implementation on xQx). In strategic oscillation the procedure is divided between two modes, it oscillates between *dropping* (changing variables 1–0) and *adding* (changing variables from 0 to 1) variables. Depending on which mode the algorithm has reached a local optimality the *flip* is implemented on a set of variables, from 2 to 12. The process is implemented (with no randomness) several cycles until some stopping criteria terminate the process. However, here in our algorithm we use some randomness to implement a form of strategic oscillation.

Note that in our algorithm we use similar short term so called, 'recency', tabu tenure structure as used in Glover et al. (1998). However, we do not use long term, 'frequency', tabu structure as they do. Furthermore, we use a simple *aspiration* criterion in our tabu search. If the objective value of a new solution is strictly better than the best known solution found so far we override the tabu status for the variable to be changed.

3. Computational experience

To test effectiveness of our procedures using multistart and *n-Opt* (*2, 3, 4-Opt* and a combination of these 3 sequential moves, *ALL*) strategies we applied the 4 procedures on three sets of benchmark problems available on the Internet that have been used by many researchers. The first set is randomly generated UBQP problems by Beasley available on http://people.brunel.ac.uk/~mastjjb/jeb/orlib/bqpinfo.html, sizes ranging from 50 to 2500 variables. However, we only used problems of sizes 1000 and 2500 in our experiments, each size includes 10 problems (total of 20 problems). Smaller size problems were solved in 0 s of CUP time

Initialization:

Set: x= a starting binary vector, $x^*=x$, $Z=Z^*=xQx$, $p1$, $p2$, $Tabu\_ten$,
$Tabu(i)=0$, for $i=1,\ldots,n$, and $\pi=(\pi_1,\ldots,\pi_n)$, calculate the set $J_{next}(x)$.

Do while (until some stopping criteria, e.g., time limit, is reached)
  Do $K=p1, p2$
    Do while ($J_{next}(x) \neq \varnothing$)
      Do $i=1, n$
        $L=\pi_i$
        If ($L \in J_{next}$) **Then**
          $\bar{Z}=\bar{x}Q\bar{x}$, for $\bar{x}_j = x_j$, $j \neq L$, and $\bar{x}_L = 1-x_L$,
          If $((Tabu(L)=0)$ or $(\bar{Z} > Z^*))$ **Then**
            $x_L = 1-x_L$,
            Update: $J_{next}(x)$, and $Tabu(j)$, *for* $j=1,\ldots,n$,
            $Last\_x\_changed = x_L$,
            $Z=\bar{Z}$,
            If $(\bar{Z} > Z^*)$ $Z^* = \bar{Z}$,
          End If
        End If
      End do
      Call *N_Opt(.)*
    End while
    Call *rand_var_change(.)*
  End do
End while

Fig. 5. Pseudo code of the simple tabu search used to test problems.

and thus we did not include in our report. The second set of problems includes 21 larger UBQP problems were generated also randomly by Palubeckis, sizes ranging from 3000 to 7000. This set of problems is available on http://www.proin.ktu.lt/~gintaras/ubqop_its.html. Detail characteristics of both sets of problems are explained in Hanafi et al. (2013). Best known solutions for both sets of problems were taken from Hanafi et al. (2013) and Lü et al. (2010). The third set of problems includes 84 maximum cut problems (Max-Cut) on graphs, sizes ranging from 125 to 3000 nodes (variables). These problems are included in two sets, Set1 and Set2, and are available via http://www.optsicom.es/maxcut/. Characteristics of each problem is explained on the given site, also in Marti et al. (2009). Note that formulating Max-Cut problem as *xQx* model is straight forward and have been used by many researchers, (see for example, Boros & Hammer, 1991; Helmberg, & Rendl, 1996; Kochenberger, Hao, Lu, Wang, & Glover, 2013; Lin & Zhu, 2014). Best known solutions for Max-Cut problems were taken from Kochenberger et al. (2013), Marti et al. (2009), Shylo and Shylo (2014).

We coded our procedures in FORTRAN and all runs were conducted on the super computer, SGI Altix XE cluster, at the University of Mississippi. We report on total of 125 *xQx* problems, Tables 3–7. In the tables the *'Objective Function Value'* represents the objective function value of the best solutions our procedures reached within the given total time. And *'Time to best'* is the CPU time an algorithm reached first the *'Objective Function Value'*.

3.1. Initial parameter settings

To solve each problem using one of procedures, *2-Opt, 3-Opt, 4-Opt*, and *ALL*, we used three starting solutions $x = 0$, $x = 1$, and a randomly generated binary solution. Furthermore, for each problem, each procedure, and each starting solution we solved the problem with two different starting sequences, one starting with a sequence π as the natural numbers, and the other in the reverse order of natural numbers. Thus, each problem was solved by each procedure six times and best results are reported. Initial computations for solving some of the problems showed the values of *Tabu_ten, p1, p2* and the total time, *'Total_time'* were the most important consideration in reaching quality solutions. The value of *Total_time* given to a problem was divided equally to each of the six ways of solving the problem by a procedure. For example, if a *2_Opt* strategy was applied to a problem six times, assuming $n = 7000$, and $Total\_time = 0.006n = 42$ s, then each of the six ways of solving the problem was given exactly $0.001n = 7$ s of CPU time.

From each set of problems and each size we randomly chose one problem (total of 14 sample problems) and solved using combination of *Tebu_ten*, and *p1, p2*, with a $Total\_Time = 0.06n$, given as follows.

Tebu_ten: 5, 10, 15, 20, 25 (total of 5 cases)
p1: 2, 5, 10, 0.01n, 0.04n, 0.07n (total of 6 cases)
p2: 10, 15, 20, 70, 90, 100, 0.025n, 0.05n, 0.10n, 0.15n, 0.20n (total of 11 cases)

Table 2
Parameters used to solve the problems.

	Beasley 20 problems	Palubeckis 21 problems	Max-Cut n <= 1000 46 problems	Max-Cut n >= 2000 38 problems
Tabu_ten	5	5	5	5
p1	0.04n	0.04n	2	2
p2	0.15n	0.15n	12	20
Total_time	0.05n (sec.)	0.5n (sec.)	0.5n (sec.)	0.5n (sec.)

Table 3
Number of times an algorithm reached the best known solution or a better solution within the given total time.

ID	Frequency			
	2_Opt	3_Opt	4_Opt	ALL
B1000.3	899	648	614	748
B2500.1	1260	811	857	861
P3000.4	4774	2488	2478	2573
P4000.1	3012	1452	1413	1507
P5000.3	189	67	76	59
P6000.3	61	22	18	29
P7000.3	861	290	294	377
G14	54	28	25	52
G22	122041	640551	807767	909922
G45	777419	1158134	616654	967347
G54	2031	971	860	650
G10400	26432	20129	20033	21567
G14500	44278	5271	5505	7514

Table 4
Computational results for 20 UBQP Beasley problems.

ID	Best known	Objective function value				Time to best (seconds)			
		2_Opt	3_Opt	4_Opt	ALL	2_Opt	3_Opt	4_Opt	ALL
b1000.1	371438	371438	371438	371438	371438	0.0	0.3	1.1	0.1
b1000.2	354932	354932	354932	354932	354932	0.1	0.2	0.1	0.2
b1000.3	371236	371236	371236	371236	371236	0.1	0.1	0.4	0.0
b1000.4	370675	370675	370675	370675	370675	0.6	1.3	1.4	0.7
b1000.5	352760	352760	352760	352760	352760	0.1	0.2	0.1	0.3
b1000.6	359629	359629	359629	359629	359629	0.1	0.8	0.1	0.2
b1000.7	371193	371193	371193	371193	371193	0.1	0.0	0.3	0.1
b1000.8	351994	351994	351994	351994	351994	0.9	0.3	0.7	0.8
b1000.9	349337	349337	349337	349337	349337	0.2	0.9	0.4	0.4
b1000.10	351415	351415	351415	351415	351415	2.9	0.2	0.1	0.5
Avg.	360460.9	360460.9	360460.9	360460.9	360460.9	0.5	0.4	0.5	0.3
b2500.1	1515944	1515944	1515944	1515944	1515944	2.8	0.9	1.7	0.5
b2500.2	1471392	1471392	1471392	1471392	1471392	5.0	8.8	4.6	2.5
b2500.3	1414192	1414192	1414192	1414192	1414192	6.0	1.7	2.4	3.5
b2500.4	1507701	1507701	1507701	1507701	1507701	0.7	0.4	0.4	0.4
b2500.5	1491816	1491816	1491816	1491816	1491816	1.0	3.3	0.5	0.5
b2500.6	1469162	1469162	1469162	1469162	1469162	2.5	1.7	1.8	1.2
b2500.7	1479040	1479040	1479040	1479040	1479040	9.2	12.0	15.0	4.5
b2500.8	1484199	1484199	1484199	1484199	1484199	13.6	0.7	1.1	2.8
b2500.9	1482413	1482413	1482413	1482413	1482413	2.3	9.6	4.5	2.2
b2500.10	1483355	1483355	1483355	1483355	1483355	4.3	3.9	16.2	1.3
Avg.	1479921.4	1479921.4	1479921.4	1479921.4	1479921.4	4.8	4.3	4.8	1.9

These numbers create 330 combinations. Thus, each of the 14 problems was solved by each of the 4 procedures about 330 times. Our initial testing took in the account possibility of some overlaps since it is needed to have *p1* < *p2*. Thus, number of initial testing is less than 330. However, our codes did not differentiate between some of the overlaps that were created by combination of *p1* and *p2* in comparison with *%n* for *p1* and *p2*. Thus, some problems may have been solved with the same parameters more than once. Our initial solutions provided some insight into parameter settings as follows. Interestingly initial tests suggested that different parameters are appropriate for the set of generated UBQP and the set of Max-Cut problems. In the next three subsections we explain parameter settings for generated problems and Max-Cut problems separately then we explain analysis of our final computational results.

3.1.1. Initial experiments for Beasley and Palubeckis UBQP problems

The two sets of generated problems include 41 problems ranging from 1000 to 7000 variables. Our initial experiments suggested that lower values of *Tabu_ten* were more suitable with all 4 procedures for solving these problems. In general for most of the problems the best values tabu tenure were about 5, thus we set *Tabu_ten* = 5 in the final experimentations. Our experiments also suggested that the results were highly sensitive to parameters,

Table 5
Computational results for 21 UBQP Palubeckis problems.

ID	Best Known	Objective Function Value				Time to best (seconds)			
		2_Opt	3_Opt	4_Opt	ALL	2_Opt	3_Opt	4_Opt	ALL
p3000.1	3931583	3931583	3931583	3931583	3931583	3.3	3.2	2.8	9.6
p3000.2	5193073	5193073	5193073	5193073	5193073	5.5	5.4	3.0	3.2
p3000.3	5111533	5111533	5111533	5111533	5111533	2.8	4.0	2.6	2.2
p3000.4	5761822	5761822	5761822	5761822	5761822	2.5	5.4	3.1	2.4
p3000.5	5675625	5675625	5675625	5675625	5675625	5.6	15.8	6.7	5.9
Avg.	**5134727.2**	**5134727.2**	**5134727.2**	**5134727.2**	**5134727.2**	**3.9**	**6.8**	**3.6**	**4.7**
p4000.1	6181830	6181830	6181830	6181830	6181830	13.6	2.5	3.8	22.9
p4000.2	7801355	7801355	7801355	7801355	7801355	68.5	2.8	22.6	5.1
p4000.3	7741685	7741685	7741685	7741685	7741685	4.1	6.5	4.0	25.9
p4000.4	8711822	8711822	8711822	8711822	8711822	1.7	18.8	1.5	1.8
p4000.5	8908979	8908979	8908979	8908979	8908979	150.8	3.1	146.1	17.0
Avg.	**7869134.2**	**7869134.2**	**7869134.2**	**7869134.2**	**7869134.2**	**47.7**	**6.7**	**35.6**	**14.5**
p5000.1	8559680	8559680	**8559355**	8559680	8559680	1478.2	2102.2	1574.4	706.5
p5000.2	10836019	10836019	10836019	10836019	10836019	289.6	866.9	579.3	18.5
p5000.3	10489137	10489137	10489137	10489137	10489137	510.8	1340.2	173.8	265.1
p5000.4	12252318	12252318	12252318	12252318	12252318	1535.1	194.3	524.4	1442.4
p5000.5	12731803	12731803	12731803	12731803	12731803	322.3	484.9	16.2	1043.1
Avg.	**10973791.4**	**10973791.4**	**10973726.4**	**10973791.4**	**10973791.4**	**827.2**	**997.7**	**573.6**	**695.1**
p6000.1	11384976	11384976	11384976	11384976	11384976	1117.9	1626.8	1386.6	1155.7
p6000.2	14333855	14333855	14333855	14333855	14333855	1190.9	1397.6	1371.2	1155.6
p6000.3	16132915	16132915	16132915	16132915	16132915	488.6	19.7	790.4	5.0
Avg.	**13950582.0**	**13950582.0**	**13950582.0**	**13950582.0**	**13950582.0**	**932.5**	**1014.7**	**1182.7**	**772.1**
p7000.1	14478676	14478676	14478676	14478676	14478676	1796.1	1333.7	1378.6	1079.0
p7000.2	18249948	18249948	**18249802**	**18249844**	**18249802**	1407.3	2978.1	2616.0	1197.7
p7000.3	20446407	20446407	20446407	20446407	20446407	87.6	17.4	24.5	22.3
Avg.	**17725010.3**	**17725010.3**	**17724961.7**	**17724975.7**	**17724961.7**	**1097.0**	**1443.1**	**1339.7**	**766.3**

p1 and *p2*. In Glover et al. (1998), these parameters for strategic oscillation were set to be 2 and 12, respectively. However, our benchmark problems are larger and different in nature compared to the problems solved by Glover et al. (1998). Furthermore, as explained earlier, their tabu search does not involve randomness in the process, and it always considers the most improving variable for the next possible flip move. In our case, for these 41 generated problems the procedures did not provide best known solutions until *p1* reached 70 and above. Furthermore, as *p2* was reaching closer to *0.15n*, the quality of the solutions were improving then started to diminish as *p2* increased. Note that in our procedures values of *p1* and *p2* basically provide opportunities for randomly flipping some variables. This change of variables allows the procedure get out of pre-mature local search. For example, if $n = 7000$, and if $p1 = 0.04 = 280$ and $p2 = 0.15n = 1050$ then we potentially are randomly flipping up to 1050 variables. This gives opportunity to expand larger search space. Although this may look unrealistic to flip so many variables, however, our extensive experiment for these sets of problems showed it worked very well when $p1 = 0.04n$ and $p2 = 0.15n$. On the issue of how much of CPU time should be given to each procedure, since we are implementing heuristics, in general more time would provide better solutions. However, to be realistic in usage of CPU time, especially, for large size problems, we decided to give at the initial experimentation $Total\_time = 0.06{*}n$ CPU time to each procedure for each problem. Furthermore, problems in the Beasley's set were quite easier to reach to the best known solutions compared to the Palubeckis's set of problems. Thus, in the final experiments for each procedure we gave *0.05n* CPU time (in seconds) to each of the Beasley's set of problems, and *0.5n* to each of the Palubeckis's set of problems.

3.1.2. Initial experiments for Max-Cut problems

In parameter settings for Max-Cut benchmark problems we had quite different experiences compared to the generated problems. Note that the smallest of these problems have 125 and the largest have 3000 variables. For the parameter *Tabu_ten* we had similar results compared to the generated benchmark problems. Thus in the final experimentation we also set $Tabu\_ten = 5$. Regarding parameters *p1* and *p2*, however, we had opposite experiences compared to the UBQP generated benchmark problems. Smaller values of *p1* and *p2* provided considerably better results compared to larger values. In fact, when *p1* reached larger than 10 a procedure rarely reached the best known solutions for any of the problems. As *p2* was increased, solution quality for all procedures diminished. Overall the best values for these two parameters that provided the best solutions for Max-Cut problems at the initial testing were $p1 = 2$, and $p2 = 10$ (also *15*) for problem sizes up to 1000, and $p1 = 2$, $p2 = 20$, for problem sizes 2000 and above. Table 2 categorizes all parameters that were set in our final experimentations.

Table 6
Computational results for SET1 Max-Cut problems.

ID	*n*	*Best Known*	*Objective Function Value*				*Time to best (seconds)*			
			2_Opt	*3_Opt*	*4_Opt*	*ALL*	*2_Opt*	*3_Opt*	*4_Opt*	*ALL*
G1	800	11624	11624	11624	11624	11624	1.6	1.4	3.8	15.9
G2	800	11620	11620	11620	11620	11620	3.7	50.7	97.5	35.3
G3	800	11620	**11622**	**11622**	**11622**	**11622**	9.0	88.7	64.2	170.1
G4	800	11646	11646	11646	11646	11646	0.2	5.0	10.8	25.9
G5	800	11631	11631	11631	11631	11631	13.7	82.2	13.6	4.3
G6	800	2178	2178	2178	2178	2178	4.0	4.9	3.3	0.5
G7	800	2006	2006	2006	2006	2006	27.7	22.3	12.5	4.3
G8	800	2005	2005	2005	2005	2005	24.7	0.3	4.2	10.3
G9	800	2054	2054	2054	2054	2054	40.8	8.7	3.4	12.8
G10	800	2000	2000	2000	2000	2000	71.7	11.5	76.4	34.4
G11	800	564	564	562	564	564	209.6	43.3	491.8	905.4
G12	800	556	556	556	556	556	25.3	56.6	12.2	68.3
G13	800	580	**582**	**582**	**582**	**582**	12.7	203.7	60.1	15.4
G14	800	3061	**3064**	3061	3061	**3064**	316.2	5.4	224.2	121.0
G15	800	3050	3050	**3049**	**3049**	**3049**	23.0	41.4	71.4	18.6
G16	800	3052	3052	3051	3051	3052	11.6	35.0	119.3	93.4
G17	800	3046	3046	3046	**3047**	3046	174.6	196.6	131.1	170.4
G18	800	991	**992**	**992**	**992**	**992**	3.8	7.8	0.8	9.3
G19	800	904	**906**	**906**	**906**	**906**	62.0	3.2	4.0	41.9
G20	800	941	941	941	941	941	17.9	9.3	2.1	5.5
G21	800	931	931	931	931	931	8.0	9.7	0.8	5.7
Avg.		**4098.1**	**4098.6**	**4098.2**	**4098.4**	**4098.5**	**50.6**	**42.3**	**67.0**	**84.2**
G22	2000	13359	13359	13359	13359	13359	46.6	109.3	4.5	19.9
G23	2000	13342	**13344**	**13344**	**13344**	**13344**	255.4	87.5	414.5	102.3
G24	2000	13337	13337	13337	13337	13337	147.4	191.1	124.6	211.4
G25	2000	13332	**13340**	**13340**	**13340**	**13339**	168.4	190.6	178.9	231.7
G26	2000	13328	13328	**13327**	13328	13328	8.2	2.9	8.5	1.9
G27	2000	3336	**3337**	**3341**	**3341**	**3341**	224.4	388.9	914.0	471.1
G28	2000	3295	**3296**	**3297**	**3298**	**3298**	174.9	204.0	272.8	300.8
G29	2000	3391	**3405**	**3405**	**3405**	**3405**	615.2	104.4	99.5	193.3
G30	2000	3403	**3409**	**3413**	**3413**	**3412**	67.3	601.2	528.4	240.3
G31	2000	3288	**3308**	**3309**	**3310**	**3310**	248.7	226.4	566.4	68.8
G32	2000	1406	**1410**	**1410**	**1410**	**1404**	197.8	189.7	121.9	529.5
G33	2000	1378	**1382**	**1382**	**1382**	**1381**	48.1	309.6	548.8	244.4
G34	2000	1378	**1384**	**1384**	**1384**	**1382**	33.6	231.9	416.5	126.4
G35	2000	7678	**7682**	**7682**	**7682**	**7682**	200.5	356.7	202.0	454.7
G36	2000	7670	7670	**7672**	**7672**	**7672**	662.0	764.0	761.0	659.0
G37	2000	7682	**7684**	7682	**7679**	**7679**	356.0	355.8	391.4	313.9
G38	2000	7683	7683	7683	7683	7683	417.6	345.7	356.4	285.1
G39	2000	2397	**2407**	**2408**	**2406**	**2405**	317.9	157.2	81.0	368.1
G40	2000	2390	**2400**	**2400**	**2397**	**2397**	544.6	170.3	621.7	31.7
G41	2000	2400	**2405**	**2405**	**2404**	**2405**	64.6	82.5	53.6	66.0
G42	2000	2469	**2480**	**2481**	**2480**	**2480**	147.7	254.8	269.6	156.4
Avg.		**6092.5**	**6097.6**	**6098.1**	**6097.8**	**6097.3**	**235.6**	**253.5**	**330.3**	**241.7**
G43	1000	6660	6660	6660	6660	6660	39.3	1.8	20.5	5.5
G44	1000	6639	**6650**	**6650**	**6650**	**6650**	54.3	32.0	43.9	16.1
G45	1000	6652	**6654**	**6654**	**6654**	**6654**	3.2	15.4	32.0	7.7
G46	1000	6649	6649	6649	6649	6649	176.2	6.8	103.7	190.7
G47	1000	6665	**6657**	**6657**	**6657**	**6657**	41.5	56.3	118.4	87.1
Avg.		**6653.0**	**6654.0**	**6654.0**	**6654.0**	**6654.0**	**62.9**	**22.5**	**63.7**	**61.4**
G48	3000	6000	6000	6000	6000	6000	0.0	8.4	0.0	0.0
G49	3000	6000	6000	6000	6000	6000	0.0	0.0	0.0	0.0
G50	3000	5880	5880	5880	5880	5880	168.2	0.0	21.0	0.0
G51	3000	3847	**3843**	**3844**	**3843**	**3844**	72.9	76.7	159.3	91.5
G52	3000	3849	**3851**	**3850**	**3851**	**3850**	92.4	45.1	63.9	51.8
G53	3000	3848	3848	**3850**	**3845**	**3845**	41.5	242.4	969.1	171.4
G54	3000	3851	**3852**	**3850**	**3852**	**3852**	141.0	159.4	117.6	81.3
Avg.		**4753.6**	**4753.4**	**4753.4**	**4753.0**	**4753.0**	**73.7**	**76.0**	**190.1**	**56.6**

Table 7
Computational results for SET2 Max-Cut problems.

ID	n	Best Known	Objective Function Value				Time to best (seconds)			
			2_Opt	3_Opt	4_Opt	ALL	2_Opt	3_Opt	4_Opt	ALL
G54100	125	110	110	110	110	110	0.0	0.1	0.0	0.0
G54200	125	112	112	112	112	112	0.0	0.0	0.0	0.0
G54300	125	106	106	106	106	106	0.0	0.0	0.0	0.0
G54400	125	114	114	114	114	114	0.0	0.0	0.0	0.0
G54500	125	112	112	112	112	112	0.0	0.0	0.0	0.1
G54600	125	110	110	110	110	110	0.0	0.0	0.0	0.0
G54700	125	112	112	112	112	112	0.0	0.2	0.0	0.1
G54800	125	108	108	108	108	108	0.0	0.1	0.1	0.1
G54900	125	110	110	110	110	110	0.0	0.0	0.0	0.0
G541000	125	112	112	112	112	112	0.0	0.0	0.0	0.0
Avg.		**110.6**	**110.6**	**110.6**	**110.6**	**110.6**	**0.0**	**0.0**	**0.0**	**0.0**
G10100	1000	894	**896**	894	894	**896**	8.8	75.6	26.0	31.3
G10200	1000	900	900	900	900	900	19.9	2.6	26.3	73.7
G10300	1000	892	892	892	892	892	26.3	239.0	152.4	46.4
G10400	1000	896	**898**	**898**	**898**	**898**	7.9	261.0	6.7	106.0
G10500	1000	882	**886**	**886**	**884**	**886**	80.2	133.7	150.5	66.8
G10600	1000	886	**888**	**888**	**888**	**888**	37.4	120.9	135.5	107.7
G10700	1000	898	898	**900**	898	898	14.6	134.3	20.1	26.5
G10800	1000	880	880	880	**882**	880	13.7	44.5	239.2	59.0
G10900	1000	900	**902**	**902**	**902**	**902**	164.4	158.8	52.2	271.9
G101000	1000	892	**894**	**894**	**894**	**894**	59.8	156.1	100.8	169.3
Avg.		**892.0**	**893.4**	**893.4**	**893.2**	**893.4**	**43.3**	**132.6**	**91.0**	**95.9**
G14100	2744	2428	**2426**	**2424**	**2426**	**2426**	439.2	257.7	449.5	579.6
G14200	2744	2424	**2444**	**2440**	**2440**	**2440**	230.7	103.9	235.9	243.3
G14300	2744	2426	**2424**	**2422**	**2424**	**2420**	639.6	211.3	167.1	657.2
G14400	2744	2426	**2434**	**2428**	**2428**	**2428**	888.7	572.8	822.0	216.4
G14500	2744	2420	**2430**	**2426**	**2426**	**2428**	338.3	422.9	957.6	538.7
G14600	2744	2426	**2434**	**2432**	**2432**	**2432**	723.3	516.8	301.8	565.4
G14700	2744	2416	**2424**	**2426**	**2424**	**2422**	401.8	461.3	798.1	171.8
G14800	2744	2422	**2432**	**2430**	**2426**	**2430**	754.3	811.2	363.7	411.2
G14900	2744	2412	**2410**	**2404**	**2406**	**2404**	872.4	460.8	589.8	224.5
G141000	2744	2430	**2442**	**2438**	**2438**	**2436**	82.1	110.8	239.3	199.4
Avg.		**2423.0**	**2430.0**	**2427.0**	**2427.0**	**2426.6**	**537.0**	**392.9**	**492.5**	**380.7**

3.1.3. Analysis of final experimentations

Using parameters given in Table 2 we solved each of 125 problems six times as explained earlier. Results are reported in Tables 3–7. However, in Table 3 we only report results of sample problems for frequencies that an algorithm reached the best known solution or better than the best known solution. In a short summary, our extensive computational results show that the procedures can reach the best known solutions with high frequencies within very short CPU time. For 123 of 125 problems the procedures reached best known solutions quickly. For the Max-Cut problems, the algorithms provided new best solutions for 44 of the 84 problems in reasonably short CPU time.

Table 3 reports number of times an algorithm reached a solution at least as good as the best known solution. Note that however some of the solutions may be the same. It is very time consuming to check if the solutions were different from each other. Thus, we only report number of times an algorithm for a problem reached the best known solution. It is clear from results of this table that different problems can have different solution space.

Table 4 shows that all algorithms reached the best known solutions for 20 Beasley's set of UBQP problem in a fraction of seconds. However, when combination of sequences were used (*ALL*) clearly average time reaching a solution was faster. Table 5 reports on Palubeckis's set of UBQP problems. For these problems also, average time to reach the best known solutions were shortest with combination of sequences, (*ALL*). However, for two problems, p5000.1, and p7000.2, the *2_Opt* strategydid not reach the best known solution within the given total time. Also, the *3_Opt* strategy and *ALL* did not reach the best known solution within the given total time. Another observation is that, for problems up to 4000 variables all strategies reached the best known solutions in very short time, ranging from 0 to 47.7 s. However, as sizes of problems grew amount of time needed to reach the best known solutions grew dramatically, ranging from 573.6 to 1443.1 s. This is consistent with frequencies reaching the best known solutions reported on Table 3.

Tables 6 and 7 report results for Max-Cut problems, SET1 and SET2. In these tables shaded cells with bold numbers provide new best solutions for the problems. Out of 84 problems the algorithms reached the best known solutions for 82 of them. For 44 of these problems the algorithms reached better than the best known solutions. However, as can be seen from the tables, *2_Opt* strategy had higher quality solutions, and this of course was with consumption of slightly more CPU time. Although the combination of all strategies (*ALL*) provided 37 new best solutions, however, quality of solutions were not as good as *2_Opt*, *3_Opt*, and *4_Opt* strategies. This is also consistent with number of problems that this strategy did not reach the best known solutions. Out of 84 Max-Cut problems, *2_Opt, 3_Opt, 4_Opt* and *ALL* strategies did not reach best known solutions within the given total time for, 4, 7, 7, and 8 problems, respectively. Bold numbers in non-shaded cells represent solutions that an algorithm did not reach the best known solution.

Regarding time to best solutions, it is difficult to reach any conclusion from the results for problems in Set1. Problems of 3000 variables were fairly easy to reach the best known solutions or solutions better than best known, however, it was more time consuming for problems of smaller sizes to reach to similar results, e.g., 2000 variables. As can be seen from Table 6, problems of 800 variables were as time consuming as problems of 3000 variables. Overall, time to reach the best solutions for problems in SET1 ranged from 0 s to 969.1 s. For problems in SET 2 however as size of problems grew time needed to reach best solutions also grew for all algorithms.

4. Conclusion

In this paper we considered the unconstrained binary quadratic program. Relationship between starting solution, the sequence of implementing local search, and the locally optimal solutions was explored. A novel diversification approach based on the sequence to implement the local improvement was proposed. We adopted limited *2, 3, 4-Opt* local optimality strategies borrowed from sequencing problems combined with multistart strategy and applied to UBQP. Extensive computational experiments for 4 strategies were conducted on set of 125 benchmark problems. Our procedures provided new best solutions for 44 of the problems.

We implemented our procedures within a simple tabu search structure. Future research can benefit from applying our procedures within more sophisticated metaheuristics. We used limited versions of *n-Opt* strategies in our procedures, however, more clever and sophisticated sequencing strategies should be explored in future research. It is well documented in the past research that multistart strategies are very powerful when applied with simple and meta-heuristics, however, in combination with sequencing improvement implementation for UBQP was new in this paper. We think future research should also concentrate on more sophisticated multistart strategies in combination with more sophisticated improvement sequencing strategies for UBQP. As we explained in Proposition 1, better starting points can reach local optimal solution faster, thus, sophisticated algorithms can benefit from exploring relationship between starting point, sequence of implementing improvement rules and the local optimal solutions. We adopted *1-flip* strategy in our procedures to reach local optimality. However, many problems, especially constrained optimization problems, benefit from *r-flip* strategies as local procedures, thus another area of possible exploration is the use of *r-flip* strategies with diversification approach presented in this paper.

Acknowledgement

The authors are grateful for comments of the referees that greatly improved presentation of the paper.

References

Ahmadi, A., Pishvaee, M. S., & Akbari Jokar, M. R. A. (2017). Survey on multi-floor facility layout problems. *Computers & Industrial Engineering, 107*, 158–170.

Aksan, Y., Dokeroglu, T., & Cosar, A. (2017). A stagnation-aware cooperative parallel breakout local search algorithm for the quadratic assignment problem. *Computers & Industrial Engineering, 103*, 105–115.

Alidaee, B., Kochenberger, G., & Wang, H. (2010). Theorems supporting r-flip search for pseudo-boolean optimization. *International Journal of Applied Metaheuristic Computing, 1*(1), 93–109.

Bayram, U., & Sahin, R. (2016). A comprehensive mathematical model for dynamic cellular manufacturing system design and Linear Programming embedded hybrid solution techniques. *Computers & Industrial Engineering, 91*, 10–29.

Boros, E., & Hammer, P. (1991). The Max-Cut problem and quadratic 0–1 optimization; polyhedral aspects, relaxations and bounds. *Annals of Operations Research, 33*, 151–180.

Boros, E., Hammer, P., & Sun, X. (1989). *The DDT method for quadratic 0–1 minimization*. RUTCOR Research Center, RRR 39–89.

Branda, M., Novotney, J., & Olstad, A. (2016). Fixed interval scheduling under uncertainty – A tabu search algorithm for an extended robust coloring formulation. *Computers & Industrial Engineering, 93*, 45–54.

Bui, T. N., & Moon, B. B. (1996). Genetic algorithm and graph partitioning. *IEEE Transactions on Computers, 45*(7), 841–855.

Chou, C.-W., Chien, C.-F., & Gen, M. (2014). A multiobjective hybrid genetic algorithm for TFT-LCD module assembly scheduling. *IEEE Transactions on Automation Science and Engineering, 11*(3), 692–705.

Das, G. S. (2017). New multi objective models for the gate assignment problem. *Computers & Industrial Engineering, 109*, 347–356.

Etscheid, M., & Roglin, H. (2014). Smoothed analysis of local search for the maximum-cut problem. *The Society for Industrial and Applied Mathematics*, 882–888.

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

Glover, F., Kochenberger, G., & Alidaee, B. (1998). Adaptive memory tabu search for binary quadratic programs. *Management Science, 44*(3), 336–345.

Glover, F., Lü, Z., & Hao, J.-K. (2010). Diversification-driven tabu search for unconstrained binary quadratic problems. *4OR-Quarterly Journal of Operational Research, 8*, 239–253.

GroSelj, B., & Malluhi, Q. M. (1995). Combinatorial optimization of distributed queries. *IEEE Transactions on Knowledge and Data Engineering, 7*(6), 915–927.

Hagen, L. W., & Kahng, A. B. (1997). Combining problem reduction and adaptive multistart: A new technique for superior iterative partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 16*(7), 709–717.

Hanafi, S., Rebai, A.-R., & Vasquez, M. (2013). Several versions of the devour digest tidy-up heuristic for unconstrained binary quadratic problems. *Journal of Heuristics, 19*, 645–677.

Hasan, M., Alkhamis, T., & Ali, J. (2000). A comparison between simulated annealing, genetic algorithm and tabu search methods for the unconstrained quadratic Pseudo-Boolean function. *Computers & Industrial Engineering, 38*, 323–340.

Helmberg, C., & Rendl, F. (1996). Solving quadratic (0–1) problems by semidefinite programs and cutting planes. Working Paper, Technische Universitat Graz.

Hwang, C.-P., Alidaee, B., & Johnson, J. D. A. (1999). Tour construction heuristic for the travelling salesman problem. *Journal of the Operational Research Society, 50*, 797–809.

James, T., Rego, C., & Glover (2009). Multistart tabu search and diversification strategies for the quadratic assignment problem. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans, 39*(3), 579–596.

Katayama, K., Tani, M., & Narihisa, H. (2000). Solving large binary quadratic programming problems by effective genetic local search algorithm. In *Proceedings of 2000 genetic and evolutionary computation conference* (pp. 643–650).

Kochenberger, G., & Glover, F. (2013). Introduction to special xQx issue. *Journal of Heuristics, 19*, 525–528.

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

Kochenberger, G., Hao, J.-K., Glover, F., Lewis, M., Lu, Z., Wang, H., et al. (2014). The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization, 28*, 58–81.

Kochenberger, G., Hao, J.-K., Lu, Z., Wang, H., & Glover, F. (2013). Solving large scale Max Cut problems via tabu search. *Journal of Heuristics, 19*, 565–571.

Kumar, P., Rosenberger, J. M., & Iqbal, G. M. D. (2016). Mixed integer linear programming approaches for land use planning that limit urban sprawl. *Computers & Industrial Engineering, 102*, 33–43.

Lewis, M., Alidaee, B., Glover, F., & Kochenberger, G. (2009). A note on xQx as a modelling and solution framework for the linear ordering problem. *International Journal of Operations Research, 5*(2), 152–162.

Lewis, M., Alidaee, B., & Kochenberger, G. (2005). Using xQx to model and solve the uncapacitated task allocation problem. *Operations Research Letters, 33*(2), 176–182.

Li, X., & Zhang, Y. (2012). Adaptive hybrid algorithms for the sequence-dependent setup time permutation flow shop scheduling problem. *IEEE Transactions on Automation Science and Engineering, 9*(3), 578–595.

Lin, G., & Zhu, W. (2014). Combining clustered adaptive multistart and discrete dynamic convexized method for the max-cut problem. *Journal of Operations Research Society of China, 2*, 237–262.

Lü, Z., Glover, F., & Hao, J.-K. (2010). A hybrid metaheuristic approach to solving the UBQP problem. *European Journal of Operational Research, 207*(3), 1254–1262.

Luo, Z., Cheang, B., Lim, A., & Zhu, W. (2013). An adaptive ejection pool with toggle-rule diversification approach for the capacitated team orienteering problem. *European Journal of Operational Research, 229*, 673–682.

Marti, M., Durante, A., & Laguna, M. (2009). Advanced scatter search for the Max-Cut problem. *INFORMS Journal on Computing, 21*, 26–38.

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

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2015). Automatic programming via iterated local search for dynamic job shop scheduling. *IEEE Transactions on Cybernetics, 45*(1), 1–14.

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

Palubeckis, G. (2006). Iterated tabu search for the unconstrained binary quadratic optimization problem. *Informatica, 17*(2), 279–296.

Pan, Q.-K., Wang, L., Sang, H.-Y., Li, J.-Q., & Liu, M. (2013). A high performing memetic algorithm for the flowshop scheduling problem with blocking. *IEEE Transactions on Automation Science and Engineering, 10*(3), 741–756.

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

Qureshi, S. A., Mirza, S. M., Rajpoot, N. M., & Arif, M. (2011). Hybrid diversification operator-based evolutionary approach towards tomographic image reconstruction. *IEEE Transactions on Image Processing*, 1977–1990.

Ren, Z., Jiang, H., Xuan, J., Hu, Y., & Luo, Z. (2014). New insights into diversification of hyper-heuristics. *IEEE Transactions on Cybernetics, 44*(10), 1747–1761.

Shi, L., & Pan, Y. (2005). An efficient search method for job-shop scheduling problems. *IEEE Transactions on Automation Science and Engineering, 2*(1), 73–77.

Shylo, V. P., & Shylo, O. V. (2014). Path relinking scheme for the max-cut problem. In Yuhui Shi (Ed.), *Emerging research on swarm intelligence and algorithm optimization*. Internet Resource, pp. -158.

Singh, S. P., & Sharma, R. R. K. (2006). A review of different approaches to the facility layout problems. *International Journal of Advanced Manufacturing Technology, 30*, 425–433.

Sun, J., Zhang, Q., & Yao, X. (2014). Meta-heuristic combining prior online and offline information for the quadratic assignment problem. *IEEE Transactions on Cybernetics, 44*(3), 429–444.

Wang, F., & Lim, A. (2008). Effective neighborhood operators for solving the flexible demand assignment problem. *IEEE Transactions on Automation Science and Engineering, 5*(2), 289–297.

Wang, Y., Lu, Z., Glover, F., & Hao, J.-K. (2012). Path relinking for unconstrained binary quadratic programming. *European Journal of Operational Research, 223*, 595–604.

Xudong, C., Guangzheng, N., & Shiyou, Y. (2002). An improved tabu algorithm applied to global optimizations of inverse problems in electromagnetics. *IEEE Transactions on Magnetics, 38*(2), 1069–1072.

Yuan, Y., & Xu, H. (2015). Multiobjective flexible job shop scheduling using memetic algorithms. *IEEE Transactions on Automation Science and Engineering, 12*(1), 336–353.

Zhou, Y., Lai, Y., & Li, K. (2015). Approximation and parameterized runtime analysis of evolutionary algorithms for the maximum cut problem. IEEE Transactions on Cybernetics, http://0-ieeexplore.ieee.org.umiss.lib.olemiss.edu/stamp/stamp.jsp?tp = &arnumber = 6897997.