Multi-wave tabu search for the boolean quadratic programming problem with generalized upper bound constraints

Zhen Shang [a], Jin-Kao Hao [c], Songzheng Zhao [b], Yang Wang [b,*]
Fei Ma [a],

[a] *School of Economics and Management, Chang'an University, Middle-section of Nan'er Huan Road, 710064 Xi'an, China*

[b] *School of Management, Northwestern Polytechnical University, 127 Youyi West Road, 710072 Xi'an, China*

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

Computers & Operation Research, November 2022
https://doi.org/10.1016/j.cor.2022.106077

Abstract

The boolean quadratic programming problem with generalized upper bound constraints (BQP-GUB) is an NP-hard problem with many practical applications. In this study, we propose an effective multi-wave tabu search algorithm for solving BQP-GUB. The algorithm performs a sequence of search waves, where each wave alternates between the forward and reverse phases, and the transition between two adjacent waves depends on a hybrid perturbation phase. The forward phase employs tabu search to reach a critical solution and the reverse phase follows to reverse previously performed moves and perform an equal number of moves by referring to the search information gathered from the latest search process. The hybrid perturbation phase randomly chooses a directed strategy, a frequency guided strategy and a recency guided strategy to achieve search diversification. Experimental results on 78 standard instances indicate that the proposed algorithm is able to improve the lower bounds for 6 instances and match the best solutions in the literature for most instances within competitive time.

Keywords: Binary quadratic programming; Tabu search; Hybrid metaheuristic; Graph theory.

* Corresponding author.
Email addresses: `sz_email@qq.com` (Zhen Shang), `jin-kao.hao@univ-angers.fr` (Jin-Kao Hao), `zhaosongzheng@nwpu.edu.cn` (Songzheng Zhao), `yangw@nwpu.edu.cn` (Yang Wang), `mafeixa@chd.edu.cn` (Fei

1 Introduction

Let $I = \{1, 2, ..., n\}$ be a finite set and c_i be a profit of $i \in I$. For each pair of element $(i, j) \in E = I \times I$, a profit q_{ij} is given. And we have a quadratic profit matrix $Q = (q_{ij})_{n\times n}$ and a linear profit vector $C = (c_1, c_2, ..., c_n)$. Suppose I is partitioned to m mutually disjoint subsets $S_1, S_2, \ldots, S_m$. Then, the boolean quadratic programming problem with generalized upper bound constraints (BQP-GUB) can be formulated as follows (Wang and Punnen, 2017):

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

$$s.t.\ \sum_{i\in S_k} x_i = 1, for\ k = 1, 2, ..., m \tag{2}$$

$$x_i \in \{0, 1\}, for\ i = 1, 2, ..., n. \tag{3}$$

A graph theoretic description can be given, if I is a vertex set, E is an edge set, and the profit of each edge (i, j) is $q_{ij} + q_{ji}$. Then BQP-GUB is to find the induced subgraph $G' = (I', E')$ of graph $G = (I, E)$ that maximizes the total profit, where I' takes one vertex, say α_k, from each S_k $(k = 1, 2, \ldots, m)$, and $E' = I' \times I'$. For example, Fig. 1 presents a graph with partitioned vertices. We choose vertices $\alpha_1 = 1, \alpha_2 = 3$ and $\alpha_3 = 5$, from subsets S_1, S_2 and S_3 to construct an induced subgraph. In this way, we need to find the induced subgraph with the maximum profit. Further, we define $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ as a solution of BQP-GUB, then the original model can be simplified as follows:

$$\max\ f(\alpha) = \sum_{t=1}^{m}\sum_{k=1}^{m} q_{\alpha_t,\alpha_k} + \sum_{k=1}^{m} c_{\alpha_k} \tag{4}$$

$$s.t.\ \alpha_k \in S_k, \text{for } k = 1, 2, \ldots, m. \tag{5}$$

From the perspective of algorithm design, the symmetric matrix is usually beneficial to the efficiency. Hence, without loss of any generality, the quadratic profit matrix Q is assumed to be symmetric, i.e., $q_{ij} = q_{ji}$ $(i, j \in I)$, in following sections. If Q is not symmetric for certain instances, Q can be replaced by symmetric $\frac{1}{2}(Q + Q^T)$, which does not affect the optimal solution set and the objective value.

BQP-GUB is a generalization of the well-studied quadratic semi-assignment problem (QSAP) that can be seen as a special case with subsets containing

Ma).

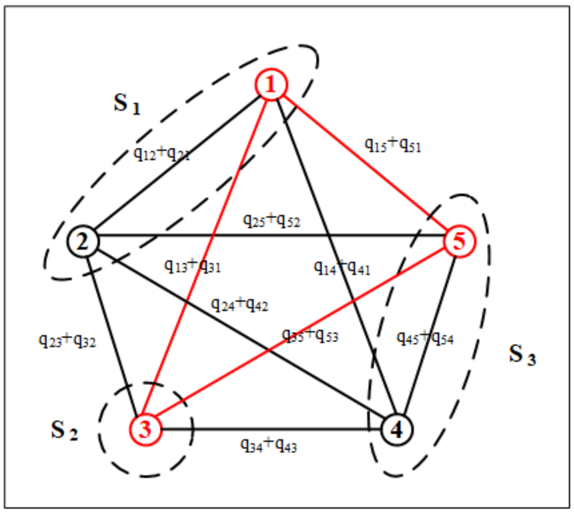


Fig. 1. An example of the BQP-GUB problem

an equal number of vertices. Hence BQP-GUB can be applied in applications of QSAP, including transit network design (Bookbinder and Désilets, 1992; Daduna and Voß, 1995), scheduling (Malucelli, 1996; Bullnheimer, 1997; Skutella, 2001), clustering (Glasner et al., 2011; Manohar et al., 2011; Duffuaa and Fedjki, 2012; Li et al., 2018) and many others. By adjusting coefficients, BQP-GUB can express the cluster-restricted maximum induced subgraph problem (MISP) and maximum clique problem (MCP), which further extends its applications to social network analysis(Pattillo et al., 2012), data mining (Cook and Holder, 2000), chemistry and biology (Raymond and Willett, 2002; Yamaguchi et al., 2004). In addition, through introducing additional variables and modifying coefficients, BQP-GUB can be recast into the unconstrained boolean quadratic programming problem (UBQP) (Lü et al., 2010; Samorani et al., 2019; Chen et al., 2020), indicating that the algorithms for UBQP can also be used to solve BQP-GUB.

Wang and Punnen (2017) have conducted systematic analyses of BQP-GUB's complexity and algorithms. This research proved that BQP-GUB is strongly NP-hard, even when $|S_k| \leq 2$, profit matrix $Q = (q_{i,j})$ is rank one and vector $C = (c_i)$ is zero. The authors reformulated BQP-GUB as UBQP, and applied the path relinking algorithm (PR1) (Wang et al., 2012), which is a state-of-the-art general algorithm for UBQP, to solve the converted model. However, experiments show that this general solver is not very effective in solving BQP-GUB. Instead, they proposed three tailored algorithms for the problem, including variable neighborhood search (VNS), iterated local search (ILS) and iterated tabu search (ITS). Computational experiments on 78 benchmark instances disclose that ITS, which is enhanced by an adaptive perturbation scheme and a historical information guided solution generation, outperforms the VNS and ILS algorithms.

Glover (2016) presented a new general multi-wave search framework, which can enhance search performance by considering candidate list, persistent attractiveness and effects of sequential decisions in memory based strategies. The whole search process consists of multiple waves, each search wave alternates between vertical phases and horizontal phases. The Vertical-Phase can be regarded as a special hill climbing approach, which performs improving moves and employs an ordered set called Active Move Record (AMR) to record performed moves by occurrence time. Since early decisions are usually made by insufficient information, the Horizontal-Phase is introduced to replace imperfect early performed moves with new moves according to AMR and the latest information, which can be considered as a kind of backtracking. This framework has been successfully used to solve challenging problems. For example, Glover et al. (2018) developed a multi-wave algorithm embracing a strategic oscillation to solve the uncapacitated facility location problem; Song et al. (2019) presented an effective multi-wave algorithm with five phases cooperating within a wave to solve the max-mean dispersion problem. Given that the forward and reverse moves are performed, we rename “Vertical-Phase” and “Horizontal-Phase” used in Glover (2016) as “Forward-phase” and “Reverse-phase”, respectively to better understand the terminology of the multi-wave algorithm.

In this study, we introduce a multi-wave tabu search algorithm (MWTS) for solving the challenging BQP-GUB problem. The proposed algorithm employs multiple memory based strategies to guide the search process, including AMR, tabu list, frequency and recency vectors. MWTS performs a sequence of search waves for intensification, where each wave consists of tabu search based forward phases and reverse phases. Between waves, the hybrid perturbation is triggered to build well-diversified solutions, which embraces a directed strategy, a frequency guided strategy and a recency guided strategy. Computational experiments on 78 benchmark instances indicate that MWTS is able to improve the lower bounds for 6 instances and match the best solutions for most instances within competitive time.

The rest of this paper is organized as follows. In Section 2, the move definitions, fast evaluation methods, and AMR are described. Section 3 presents the general framework and the main components of the proposed algorithm. Section 4 reports the experimental results and comparisons with a state-of-the-art algorithm in literature. Section 5 conducts systematic analyses for key parameters and ingredients of our algorithm on the performances. Finally, Section 6 draws conclusions.

2 Basic definitions

This section presents the fundamental ingredients of our algorithm. We first define the concepts of the forward and reverse moves. Then the fast evaluation methods is proposed to calculate the objective gains of performing moves. Finally, we introduce the key ingredient of the multi-wave framework, i.e., Active Move Record (AMR) and its updating method.

2.1 Forward move and reverse move

"Forward move" and "Reverse move" are fundamental components of the multi-wave search framework for solution modification (Glover, 2016). Since the general framework is proposed for a variety of scenarios, these concepts focus on the general role of moves in the search process, rather than specific operation details. The "Forward" terminology refers to constructive moves for constructive algorithms or improving moves for neighborhood search algorithms. The "Reverse" terminology represents the moves that destruct the current solution or cancel previous performed moves. For example, in a binary optimization problem, a forward move sets x_i as 0 or 1, then the corresponding reverse move assigns $x_i = 1 - x_i$, which eliminates the effect of a performed move.

For the BQP-GUB problem, the proposed MWTS algorithm is a neighborhood search algorithm. Hence, we define the forward move as such a move that can lead to an improved solution by assigning a new vertex to the corresponding solution component, and the total number of neighbor solutions is $n-m$. Given a solution $\alpha = (\alpha_1, \alpha_2, ..., \alpha_m)$, forward move (α_k, i) consists in assigning the vertex $i \in S_k$ to a solution component α_k to convert the solution α into its neighbor solution α', which leads to an improvement of the current solution. The forward move essentially swaps the new and old vertices for a solution component by overwriting the value of α_k. In addition, since the proposed algorithm employs tabu search strategy, in each search iteration, the algorithm selects the best move as forward move to improve the current solution, based on a tabu rule and an aspiration rule. Once a forward move is performed, the algorithm records it in the Active Move Record (AMR) for subsequent modifications. Further, we provide the formal definition of the reverse move as follows.

Correspondingly, the reverse moves of (α_k, i) consist of all such candidate moves (α_k, i') satisfying $i' \in S_k, i' \neq i$, which can eliminate the effect of move (α_k, i). Once a reverse move is performed, the involved forward move is canceled, i.e., the record from AMR is removed. The definitions show that both kinds of moves overwrite solution components for vertices exchange. However, they play different roles in a search wave and trigger different operations on AMR.

2.2 Move evaluation method

An efficient move evaluation method is essential for the algorithm to quickly find high-quality moves. For BQP-GUB problem, Anacleto et al. (2021) proposed two closed-form formulae to accelerate the process of evaluating r-flip moves, which are applicable to binary solutions. In this study, we use the change of objective value (objective gain) as a direct quality indicator of forward and reverse moves, however the time complexity of applying Equation (4) is $O(m^2)$. Hence, we introduce an incremental streamlining method for move evaluation, which is a standard approach for various algorithms, such as: Wang and Punnen (2017); Song et al. (2019).

Assume that $\alpha_k = i$ is a component of the solution α. If the move (α_k, i') is performed on α, then the vertex i is replaced by the vertex i'. The exchange between two vertices causes the change of the objective value. Based on this, we define $C_{i'}(\alpha)$ and $C_i(\alpha)$ as the objective differences caused by the vertex i' and the vertex i respectively, which are calculated as follows.

$$C_{i'}(\alpha) = \sum_{t=1,t\neq k}^{m} 2q_{i'\alpha_t} + 2q_{i'i'} + c_{i'}, i' \notin \alpha, i' \in S_k \tag{6}$$

$$C_i(\alpha) = -\sum_{t=1,t\neq k}^{m} 2q_{i\alpha_t} - 2q_{ii} - c_i, i \in \alpha, i \in S_k \tag{7}$$

According to Equations (6)-(7), the total objective gain of the move (α_k, i') is $C_{i'}(\alpha) + C_i(\alpha)$, which can be used to evaluate forward and reverse moves. Further, the resulting objective value $f(\alpha')$ after performing the move (α_k, i') can be easily calculated according to Equation (8).

$$f(\alpha') = f(\alpha) + C_{i'}(\alpha) + C_i(\alpha) \tag{8}$$

Once the current solution α is updated to α', for any vertex $j \in S_t$ $(t = 1, 2, ..., m)$, $C_j(\alpha)$ can be updated to $C_j(\alpha')$ through Equations (9)-(12), which is supposed to help achieve the objective gain in the next iteration.

$$C_j(\alpha') = C_j(\alpha) - 2q_{ji} + 2q_{ji'}, (t \neq k, \alpha_t \neq j) \tag{9}$$

$$C_j(\alpha') = C_j(\alpha) + 2q_{ji} - 2q_{ji'}, (t \neq k, \alpha_t = j) \tag{10}$$

$$C_j(\alpha') = -C_j(\alpha), (t = k, j = i' or j = \alpha_k) \tag{11}$$

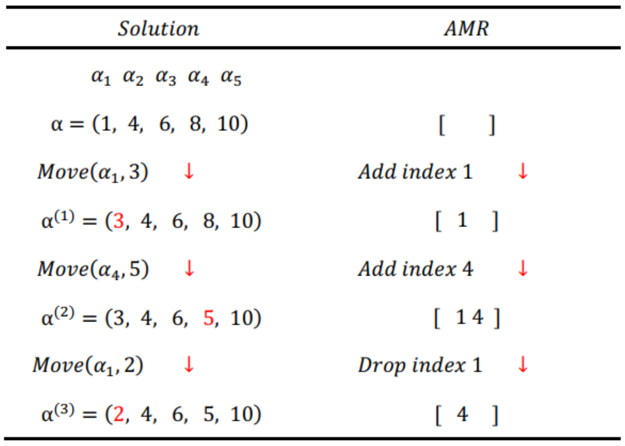
$$C_j(\alpha') = C_j(\alpha), (t = k, j \neq i', j \neq \alpha_k) \tag{12}$$

The above methods show that the time complexities of calculating resulting objective value and objective gain are reduced to $O(1)$, and updating $C_j(\alpha)$ for all vertices takes $O(n)$, which means that the incremental streamlining based approach can significantly boost the search efficiency.

2.3 Active move record

The Active Move Record (AMR) is the structure used to record forward moves that have been performed and not cancelled yet by reverse moves. During the whole search process, AMR is applied and maintained as a queue where the earlier moves precede later ones. For a move (α_k, i), we record its index k in AMR. In order to assure the feasibility of solutions, a move and its reverse move can not be simultaneously recorded in AMR. Hence, for any move (α_k, i) in AMR, if its reverse move (α_k, i') is performed, we drop the move (α_k, i) by deleting the index k from AMR and ignore reverse move (α_k, i'), which leads to a decrease in the length of AMR.

Fig. 2 presents a simple example to illustrate how AMR is updated. At first, α is an initial solution and AMR is an empty queue. The forward moves $(\alpha_1, 3)$ and $(\alpha_4, 5)$ are successively performed to transfer the solution α to $\alpha^{(2)}$. Meantime, the indexes 1 and 4 are added to the end of AMR. Then the forward move $(\alpha_1, 2)$ is performed to generate the solution $\alpha^{(3)}$. Since move $(\alpha_1, 2)$ is the reverse move of move $(\alpha_1, 3)$, the index 1 is dropped from AMR and the length of AMR decreases.



Solution	AMR
$\alpha_1\ \alpha_2\ \alpha_3\ \alpha_4\ \alpha_5$	
$\alpha = (1,\ 4,\ 6,\ 8,\ 10)$	[ ]
$Move(\alpha_1, 3)$ ↓	*Add index* 1 ↓
$\alpha^{(1)} = (3,\ 4,\ 6,\ 8,\ 10)$	[1]
$Move(\alpha_4, 5)$ ↓	*Add index* 4 ↓
$\alpha^{(2)} = (3,\ 4,\ 6,\ 5,\ 10)$	[1 4]
$Move(\alpha_1, 2)$ ↓	*Drop index* 1 ↓
$\alpha^{(3)} = (2,\ 4,\ 6,\ 5,\ 10)$	[4]

Fig. 2. An example of the AMR update

AMR essentially records the sequential decisions of optimization in the perspective of occurrence time. Different from the tabu list, AMR stores all the performed forward moves (not cancelled) instead of the moves that are prohibited within a certain period of time. Since the earlier decisions are usually made based on incomplete information (Glover, 2016; Song et al., 2019), with the help of the AMR, it is possible for the algorithm to remove the earlier forward moves and perform new high-quality moves according to the latest search information.

3 Multi-wave tabu search

In this section, we discuss the details of the proposed multi-wave tabu search (MWTS), including the main scheme, the parameter initialization method, the tabu search based forward phase, the reverse phase and the hybrid perturbation phase that randomly performs three different perturbation strategies.

3.1 Main scheme

Algorithm 1 shows the main scheme of the proposed algorithm. Starting with a random initial solution, MWTS performs multiple search waves until meeting a given stopping criterion (lines 5-29). During each wave, the algorithm first initializes key parameters *hspot* and *hspace*, which determine the timing of search waves switching from forward phases to reverse phases (line 10, Sect.3.2). Then a tabu search based forward phase repeatedly performs forward moves to improve the current solution along with AMR updating (line 12, Sect.3.3). Once the length of AMR reaches an interrupting point *hspot*, the forward phase stops, and the reverse phase continues to correct the imperfect parts of the current solution by cancelling early performed moves recorded in AMR, and performing an equal number of forward moves according to the latest search information (line 19, Sect.3.4). The resulting solution is sent to the next forward phase for further refinement. These two phases are alternately performed until the number of cycles reaches parameter hn. Finally, based on the current search status, the algorithm adaptively selects a hybrid oscillation or a random restart procedure to produce a new starting solution for the next wave (lines 23-28, Sect.3.5).

Algorithm 1 Main Scheme of Multi-wave tabu search algorithm

```
1: Input: $Q, S_1, S_2, ..., S_m$.
2: Output: The best found solution $\alpha^*$ and its objective value $f^*$.
3: Construct a random initial solution $\alpha$ and calculate its objective value $f$.
4: Set $\alpha^* = \alpha, f^* = f$, $NonImp = 0$.
5: **repeat**
6:    Set $AMR = \emptyset, Fre = \emptyset, Rec = \emptyset$.
7:    /*Search wave begins*/
8:    **for** $i$ from 1 to $hn + 1$ **do**
9:       **if** $i = 1$ **then**
10:         $(hspot, hspace) \leftarrow$ Parameters-Initialization().
11:      **end if**
12:      $(\alpha, f, \alpha^{(h)}, hspot, AMR, Rec, Fre) \leftarrow$ Forward-Phase($\alpha, f, hspot, AMR$).
13:      **if** $f > f^*$ **then**
14:         $\alpha^* = \alpha, f^* = f$, $NonImp = 0$.
15:      **else**
16:         $NonImp = NonImp + 1$.
17:      **end if**
18:      **if** $i \leq hn$ **then**
19:         $(\alpha, f, AMR) \leftarrow$ Reverse-Phase($\alpha^{(h)}, AMR, Fre$).
20:      **end if**
21:   **end for**
22:   /*Search wave terminates*/
23:   **if** $NonImp < ns$ **then**
24:      $(\alpha, f) \leftarrow$ Hybrid-Perturbation($\alpha, Rec, Fre$).
25:   **else**
26:      $(\alpha, f) \leftarrow$ Random-Restart().
27:      $NonImp = 0$.
28:   **end if**
29: **until** a stopping criterion is satisfied
```

3.2 Parameters initialization

Parameters *hspot* and *hspace* are key parameters for the algorithm to dispatch the forward phase and the reverse phase during each search wave. They are closely related to AMR and need to be initialized before each wave begins. Parameter *hspot* is a mark on the length of AMR, which is a signal for the algorithm to stop the forward phase and start the reverse phase. Parameter *hspace* is the step size for *hspot* updating. When the forward phase finishes, *hspot* is updated by $hspot = hspot + hspace$ for the algorithm to control the next forward phase.

We determine the initial values of *hspot* and *hspace* by using an estimate length of the final AMR when the local optimum is achieved. Before a search wave begins, the algorithm performs a preliminary forward phase without any interruption. When this forward phase terminates, the algorithm records the length L of AMR, which indicates the rough number of performed forward moves required for obtaining the local optimum. Then we set $hspace = L/hn$ where hn is a given parameter implying the number of cycles for the forward phase and the reverse phase during a wave, and set $hspot = hspace$ for the first forward phase in the current wave. After the initialization procedure finishes, *hspot* continues to be updated for the next forward phase and *hspace* keeps unchanged.

3.3 Forward phase

The tabu search based forward phase is the key intensification component of the multi-wave search framework, which also has the capabilities of updating AMR and collecting useful information for the following phases. The details are presented in Algorithm 2.

The objective gain indicates the change of objective value caused by a performed move. For maximization problems, moves with a higher objective gain make the objective value increase faster. Hence, the forward phase uses the objective gain $C_{i'}(\alpha) + C_i(\alpha)$ to evaluate the quality of the forward moves. This phase continually performs high-quality forward moves to improve the current solution. Specifically, all the forward moves are classified as tabu moves and non-tabu moves according to a tabu rule and an aspiration rule (Ma et al., 2017; Shang et al., 2019; Chang et al., 2021). The tabu rule demands that a recently performed forward move can not be reversed for the next t iterations. Hence, a tabu list TL is applied to record the prohibited iterations when a forward move is performed. Assume that in the *iter* iteration, a forward move on $\alpha_k = i$ replaces the vertex i with the vertex j in the current solution, then we set $TL(i) = iter + t$ and $TL(j) = iter + t$ to forbid the two vertices from exchanging for the following t iterations. We determine a move (α_k, j) as tabu if $iter \leqslant TL(\alpha_k) \bigcap iter \leqslant TL(j)$, otherwise as non-tabu. Furthermore, an aspiration rule is utilized to override the tabu rule, if a tabu forward move is able to generate a solution that is better than the best solution found so far.

Algorithm 2 Procedures for Forward Phase

```
1: Input: $\alpha, f, hspot, AMR$.
2: Output: $\alpha, f, \alpha^{(h)}, hspot, AMR, Rec, Fre$.
3: Set $\alpha^{(b)} = \alpha, f^{(b)} = f, N = 0, iter = 0, flag = 0$.
4: while $N < nt$ do
5:     (move($\alpha_k, i'$),$\alpha, f$) $\leftarrow$ Perform Best-NonTabu or Aspiration-Move().
6:     Update tabu list ($TL$, move($\alpha_k, i'$)).
7:     if $f(\alpha) > f(\alpha^{(b)})$ then
8:         $\alpha^{(b)} = \alpha, f^{(b)} = f,\ N = 0$.
9:     else
10:        $N = N + 1$.
11:    end if
12:    $Fre(i') = Fre(i') + 1$.
13:    $Rec(i') = iter$.
14:    if $flag = 0$ then
15:        Add $k$ to the end of AMR.
16:    end if
17:    if $|AMR| = hspot$ then
18:        Set $\alpha^{(h)} = \alpha,\ flag = 1,\ hspot = hspot + hspace$.
19:    end if
20:    $iter = iter + 1$.
21: end while
22: Set $\alpha = \alpha^{(b)}, f = f^{(b)}$.
```

Different from the typical tabu search (Glover et al., 2007; Lü and Hao, 2010), the forward phase employs AMR to record the execution order of forward moves. When the length of AMR reaches the interrupting point *hspot*, the current solution α is saved as $\alpha^{(h)}$ waiting for further adjustment by the reverse phase, and the AMR updating stops. According to Glover (2016), the interrupting point is a signal for finishing the forward phase. However, since the tabu search has been running for numerous iterations, in order to check the local optimums as many as possible, we let the tabu search continue without AMR until the continuous non-improvement number of iterations N reaches a given cutoff nt. Finally, the algorithm terminates the current forward phase and sets $hspot = hspot + hspace$ for the next forward phase.

In addition, the forward phase employs the vectors Fre and Rec to collect valuable information during the search process so as to guide the reverse phase and the hybrid perturbation phase. Specifically, the vector Fre records the frequency of each vertex involved in the performed moves, and the vector Rec records the latest iteration when a vertex is selected. For any vertex i, the initial values of $Fre(i)$ and $Rec(i)$ are 0. Assume that the vertex i is assigned to α_k in the $iter$ iteration, then $Fre(i) = Fre(i) + 1$ and $Rec(i) = iter$. The above information is essential for the subsequent phases, since a vertex with higher Fre value may contribute more to the final solution, and a higher Rec value indicates that the neighborhood of the vertex has been explored recently.

3.4 Reverse phase

According to the principle of the marginal conditional validity (MCV principle), due to the lack of information, the forward moves that perform earlier may not be good decisions (Glover, 2000, 2016). As the search process moves on, the search information becomes richer and more reliable, it is worthwhile to modify the previous performed moves based on the latest search information. Hence we design the reverse phase to adjust the solution $\alpha^{(h)}$ by replacing the forward moves in AMR. The details are presented in Algorithm 3.

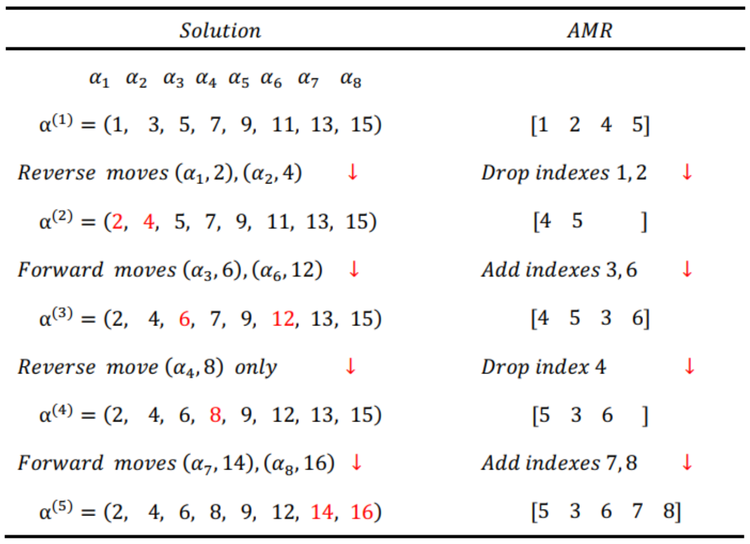
The reverse phase is composed of the dropping operation and the updating operation, which are alternately performed until all the early moves recorded in AMR are cancelled and replaced with new forward moves. The dropping operation cancels d forward moves from the beginning of AMR by performing the corresponding reverse moves. To be specific, we choose a vertex $i \in S_k/\{\alpha_k^{(h)}\}$ with the highest $C_i(\alpha^{(h)})$ as the reverse move to cancel its forward move in AMR. In addition, we introduce a vector Fre based vertex fixation strategy to avoid those vertices that obtain high Fre values from being dropped. If a vertex i is assigned to $\alpha_k^{(h)}$ for many times in the previous search process, then this vertex is assessed as a good option for the current solution. Accordingly, for any vertex i with $Fre(i) \geq \rho$, the algorithm maintains its corresponding index k in AMR and fixes the vertex i in the current solution.

After dropping d forward moves, the updating operation is to add d forward moves to the end of AMR. For each move to be added, we first select clh forward moves with the highest move gain $C_{i'}(\alpha^{(h)}) + C_i(\alpha^{(h)})$ into a candidate list, from which one move is randomly performed. In addition, we require that the selected forward move is not the reverse move of any performed moves in AMR and its Fre value is not higher than ρ, which leads the updating operation to choose the forward moves that have higher qualities but fewer chances to be selected.

Algorithm 3 Procedures for Reverse Phase

```
1: Input: $\alpha^{(h)}, AMR, Fre$.
2: Output: $\alpha, f, AMR$.
3: Set $n = 0$.
4: **repeat**
5:     **repeat** /*Dropping operation*/
6:         Get $k$ from the front of AMR and $\alpha_k^{(h)} = i$ from solution $\alpha^{(h)}$.
7:         **if** $Fre(i) < \rho$ **then**
8:             $n = n + 1$.
9:             $(\alpha^{(h)}, f^{(h)}) \leftarrow$ Perform the reverse move$(\alpha_k^{(h)}, i')$ with the highest
               $C_{i'}(\alpha^{(h)})$.
10:            $Fre(i') = Fre(i') + 1$.
11:            Delete $k$ from AMR .
12:        **end if**
13:        Check the next index in AMR.
14:    **until** $n = d$
15:    **repeat** /*Updating operation*/
16:        $n = n - 1$.
17:        Select $clh$ moves with highest move gain into the candidate list.
18:        /* These moves are not the reverse moves of those in AMR and their
           $Fre$ values are under $\rho$. */
19:        $(\alpha^{(h)}, f^{(h)}) \leftarrow$ Randomly select move$(\alpha_k^{(h)}, i')$ from the candidate list
           and perform it.
20:        $Fre(i') = Fre(i') + 1$.
21:        Add $k$ to the end of AMR.
22:    **until** $n = 0$
23: **until** All moves in AMR have been modified
24: Set $\alpha = \alpha^{(h)}, f = f^{(h)}$.
```

Fig. 3 illustrates the process of the reverse phase. Assume that $\alpha^{(h)} = (1, 3, 5, 7, 9, 11, 13, 15)$, $AMR = [1, 2, 4, 5]$, $d = 2, \rho = 3$. At first, the dropping operation performs the reverse moves $(\alpha_1, 2), (\alpha_2, 4)$ to cancel forward moves $(\alpha_1, 1), (\alpha_2, 3)$, and drop the indexes $1, 2$ from AMR. Meanwhile the updating operation performs two qualified moves $(\alpha_3, 6), (\alpha_6, 12)$, and adds the indexes $3, 6$ into the AMR. Then, the dropping operation checks forward moves $(\alpha_4, 7), (\alpha_5, 9)$ and finds the $Fre(9) > 3$. Hence only the reverse move $(\alpha_4, 8)$ is performed with index 4 dropped. The vertex 9 and the index 5 are kept unchanged. Finally. the updating operation performs forward moves $(\alpha_7, 14), (\alpha_8, 16)$, and adds the indexes $7, 8$ into AMR. Since AMR is renovated, the reverse phase terminates.



Solution	*AMR*
α_1 α_2 α_3 α_4 α_5 α_6 α_7 α_8	
$\alpha^{(1)} = (1, 3, 5, 7, 9, 11, 13, 15)$	[1 2 4 5]
Reverse moves $(\alpha_1, 2), (\alpha_2, 4)$ ↓	*Drop indexes* 1, 2 ↓
$\alpha^{(2)} = (2, 4, 5, 7, 9, 11, 13, 15)$	[4 5]
Forward moves $(\alpha_3, 6), (\alpha_6, 12)$ ↓	*Add indexes* 3, 6 ↓
$\alpha^{(3)} = (2, 4, 6, 7, 9, 12, 13, 15)$	[4 5 3 6]
Reverse move $(\alpha_4, 8)$ *only* ↓	*Drop index* 4 ↓
$\alpha^{(4)} = (2, 4, 6, 8, 9, 12, 13, 15)$	[5 3 6]
Forward moves $(\alpha_7, 14), (\alpha_8, 16)$ ↓	*Add indexes* 7, 8 ↓
$\alpha^{(5)} = (2, 4, 6, 8, 9, 12, 14, 16)$	[5 3 6 7 8]

Fig. 3. An example of the reverse phase

3.5 Hybrid perturbation phase

Since BQP-GUB is a generalization of several different problems, MWTS may encounter various situations in practical use, and it is difficult to determine an appropriate perturbation approach to enhance the search diversification. Hence, we design a hybrid perturbation phase that introduces a directed strategy, a frequency guided strategy and a recency guided strategy. After a search wave terminates, the hybrid perturbation phase randomly performs one strategy to create a diversified initial solution by changing nearly 40% of the current solution. The details are described as follows.

The directed strategy creates a diversified new solution by continually performing high-quality forward moves on the best solution generated by the previous search wave. Specifically, in each perturbation iteration, the directed strategy selects top *clp* forward moves with the highest move gains into a candidate list, from which one forward move is chosen to be performed.

The frequency guided strategy is dedicated to drive the search wave to an unexplored distant region. Generally speaking, the vertices with fewer opportunities involving in the performed moves are usually related to such solutions that lie in the neighborhoods not fully explored. Recall that the vector Fre holds the frequency information of each vertex being selected during a search wave. Hence, according to the vector Fre, in each perturbation iteration, this strategy chooses the vertices with the minimum Fre values to yield the diversified initial solution.

The recency guided strategy aims to prevent the search wave from getting trapped into a local optimum. Revisiting the recently selected vertices may lead the search wave to the same neighborhoods where the local optimum has already been found. The vector *Rec* stores the latest iteration number when a vertex is selected. The smaller the *Rec* value is, the earlier the vertex is visited. Accordingly, considering the vector *Rec*, in each perturbation iteration, this strategy selects the vertices with the minimum *Rec* values to generate the new initial solution.

4 Computational experiments

This section presents computational experiments to assess the performance of the MWTS algorithm. Specifically, we first introduce the benchmark instances, experimental protocol and recommended parameter settings. Then we make a comparison between the MWTS algorithm and the state of the art algorithm from Wang and Punnen (2017).

4.1 Benchmark instances and experimental protocol

Four benchmark sets with a total of 78 instances from Wang and Punnen (2017) are used in the experiments. The characteristics of these instances are described as follows:

- ORLIB benchmark: 30 UBQP instances from ORLIB, where n ranges from 500 to 2500, $m = 0.1 * n$, q_{ij} is a random integer ranging from -100 to 100.
- Palubeckis benchmark: 18 UBQP instances from Palubeckis, where n ranges from 3000 to 6000, $m = 0.1 * n$ and q_{ij} is a random integer ranging from -100 to 100.
- CMIS benchmark: 15 Clustered Max Induced Subgraph instances, where n ranges from 1000 to 5000, m is various, and if edge (i, j) exists, q_{ij} is a random integer ranging from 0 to 100, otherwise $q_{ij} = 0$.
- CMC benchmark: 15 Clustered Max Clique instances, where n ranges from 1000 to 5000, m is various, and if edge (i, j) exists, q_{ij} is a random integer ranging from 0 to 10, otherwise q_{ij} equals a very small negative number M.

The MWTS algorithm is programmed in C++ and compiled by GNU g++ on a computer running Linux with an Intel Xeon E5440 2.83GHz processor and 8GB RAM. The time limits for instances with 500, 1000 and 2500 vertices are set at 1, 2, and 10 minutes, and for the larger instances with 3000, 4000, 5000 and 6000 vertices are set to be 1, 1, 2 and 4 hours respectively (same as Wang and Punnen (2017)). Given the stochastic nature of the MWTS algorithm, each instance is solved 20 times independently. Table 1 provides the recommended parameter settings for the proposed algorithm, and the detailed parameter analyses are presented in Sect. 5.1.

Table 1
Settings of parameters

Parameters	Section	Description	Values
ns	3.1	Maximum continuous non-improvement iterations to restart the search procedure	5
nt	3.3	Maximum continuous non-improvement iterations to terminate the tabu search	$5 * n$
t	3.3	Tabu tenure	$0.01 * (n - m)$
clp	3.5	The size of candidate move list in the hybrid perturbation phase	20
hn	3.2	The number of cycles for the forward and reverse phases during a wave	5
clh	3.4	The size of candidate move list in the reverse phase	2
ρ	3.4	Maximum vertex selected times to maintain a vertex in the solution	5

4.2 Computational comparisons

This section provides detailed experimental comparisons between the proposed MWTS algorithm and the best ITS algorithm from Wang and Punnen (2017). Both algorithms run on the same computer and follow the same experimental protocol. The two-tailed sign test (Demšar, 2006; Zhou et al., 2020) is applied to determine if there exists a statistical significant performance difference between two algorithms. For a pairwise comparison with N instances, if an algorithm wins at least $CV_{0.05}^{N} = N/2 + 1.96\sqrt{N}/2$ times, then it is the overall winner with a significant level of 0.05. In addition, the number of tied matches is counted by splitting it equally between two algorithms.

Tables 2-5 summarize the computational results of MWTS and ITS on ORLIB, Palubeckis, CMIS and CMC benchmarks respectively. In each table, Columns 1 and 2 present the name of each instance and the best known result (BKR) in the literature. Columns marked by MWTS and ITS report the best result gap ($Gap_{bst} = f_{bst}-$BKR) between the best objective value (f_{bst}) and BKR, the average result gap ($Gap_{avg} = f_{avg}-$BKR) between the average objective value (f_{avg}) and BKR, and the average time in seconds to reach the best objective value.

Table 2 indicates that both MWTS and ITS are able to reach the BKR values for all ORLIB instances. In terms of the average objective value, MWTS attains 3 better, 17 equal and 0 worse results compared to ITS. However, since ORLIB instances are easy for both algorithms, MWTS performs marginally better than ITS on this benchmark set ($11 < CV_{0.05}^{30} \approx 20$). Table 3 shows that none of these algorithms can improve the BKR values of Palubeckis instances. With respect to the average objective value, MWTS performs significantly better than ITS with 12 better, 3 equal and 3 worse results ($13 = CV_{0.05}^{18} \approx 13$).

These results demonstrate the stability of MWTS for solving ORLIB and Palubeckis instances.

The performance differences between MWTS and ITS are more evident when solving CMIS and CMC instances. From Table 4, we observe that MWTS performs better than ITS by improving the BKR value for 1 CMIS instance. For the average objective value, MWTS performs slightly better than ITS with 6 better, 8 equal and 1 worse results ($10 < CV_{0.05}^{15} \approx 11$). From Table 5, we find that MWTS is able to improve the lower bounds for 5 CMC instances, and significantly outperforms ITS in terms of the average objective value (10 better and 5 equal, $12 > CV_{0.05}^{15} \approx 11$). To conclude, these results confirms that the multi-wave framework enables the tabu search to achieve high-quality solutions more efficiently than the typical ITS using similar computational resources.

Table 2
Computational results of MWTS and ITS for ORLIB benchmark

Instances	BKR	MWTS			ITS		
		Gap_{bst}	Gap_{avg}	time	Gap_{bst}	Gap_{avg}	time
ORLIB500-50-1	18547	0	0	0.44	0	0	0.64
ORLIB500-50-2	19529	0	0	0.03	0	0	0.09
ORLIB500-50-3	18797	0	0	0.50	0	0	0.52
ORLIB500-50-4	19193	0	0	0.08	0	0	0.12
ORLIB500-50-5	19705	0	0	0.02	0	0	0.06
ORLIB500-50-6	18772	0	0	0.25	0	0	0.30
ORLIB500-50-7	19597	0	0	0.04	0	0	0.10
ORLIB500-50-8	19616	0	0	0.05	0	0	0.13
ORLIB500-50-9	19224	0	0	0.05	0	0	0.10
ORLIB500-50-10	19234	0	0	0.06	0	0	0.11
ORLIB1000-100-1	53012	0	0	3.44	0	0	3.44
ORLIB1000-100-2	53097	0	0	5.67	0	0	10.30
ORLIB1000-100-3	53741	0	0	2.28	0	0	2.91
ORLIB1000-100-4	53578	0	0	1.85	0	0	2.79
ORLIB1000-100-5	52514	0	0	17.26	0	0	12.36
ORLIB1000-100-6	53499	0	0	2.54	0	0	2.17
ORLIB1000-100-7	53535	0	0	5.28	0	0	13.81
ORLIB1000-100-8	53721	0	0	0.80	0	0	0.63
ORLIB1000-100-9	53158	0	0	1.65	0	0	1.87
ORLIB1000-100-10	52754	0	0	1.37	0	0	1.00
ORLIB2500-250-1	209965	0	0	245.43	0	-34.50	191.84
ORLIB2500-250-2	209369	0	0	62.01	0	0	109.95
ORLIB2500-250-3	208700	0	0	107.67	0	-8.25	126.13
ORLIB2500-250-4	209761	0	0	121.88	0	0	209.42
ORLIB2500-250-5	211227	0	0	46.68	0	0	54.02
ORLIB2500-250-6	209280	0	0	104.34	0	0	241.12
ORLIB2500-250-7	211233	0	0	112.01	0	0	126.09
ORLIB2500-250-8	209380	0	0	340.67	0	-3.75	172.61
ORLIB2500-250-9	210425	0	0	74.19	0	0	86.62
ORLIB2500-250-10	210373	0	0	83.33	0	0	151.14

Table 3
Computational results of MWTS and ITS for Palubeckis benchmark

Instances	BKR	MWTS			ITS		
		Gap_{bst}	Gap_{avg}	time	Gap_{bst}	Gap_{avg}	time
P3000.1	603303	0	0	391.25	0	0	897.79
P3000.2	765619	0	0	1107.51	0	-177.65	1375.79
P3000.3	770868	0	0	707.47	0	0	1050.06
P3000.4	858994	0	0	558.92	0	-872.55	740.06
P3000.5	855323	0	0	1380.59	0	0	757.87
P4000.1	935001	0	-1.10	1475.21	0	-22.05	1593.67
P4000.2	1179507	0	-117.85	1998.33	0	-386.05	1506.31
P4000.3	1180881	0	-7.25	1408.47	0	-142.55	1306.00
P4000.4	1315725	0	-294.45	1856.91	0	-879.65	959.56
P4000.5	1331694	0	-55.60	956.37	0	-278.00	1308.01
P5000.1	1303741	0	-445.35	4193.69	0	-659.40	3294.83
P5000.2	1649346	0	-1747.75	4074.21	0	-1290.20	3308.75
P5000.3	1641435	0	-85.05	3964.01	0	-118.35	3029.14
P5000.4	1848641	0	-1470.00	3365.41	0	-741.45	3975.82
P5000.5	1854488	0	-733.80	3405.77	0	-767.35	2557.94
P6000.1	1713855	0	-2664.35	7744.10	0	-3165.50	8529.13
P6000.2	2174890	0	-1397.30	7012.88	0	-380.90	6125.35
P6000.3	2434187	0	-3090.35	7799.52	0	-3169.40	9161.35

Table 4
Computational results of MWTS and ITS for CMIS benchmark

Instances	BKR	MWTS			ITS		
		Gap_{bst}	Gap_{avg}	time	Gap_{bst}	Gap_{avg}	time
CMIS1000-20	22291	0	0	1.29	0	0	1.59
CMIS1000-50	103519	0	0	3.17	0	0	5.35
CMIS1000-100	346271	0	0	1.98	0	0	1.82
CMIS1000-200	1210755	0	0	2.28	0	0	2.33
CMIS1000-400	4372252	0	0	0.82	0	0	1.04
CMIS3000-50	111833	0	-20.35	1181.61	0	-137.10	1017.30
CMIS3000-100	373454	0	-23.45	1211.64	0	-61.75	1662.39
CMIS3000-200	1302202	0	0	1004.7	0	0	433.00
CMIS3000-500	7137065	0	0	524.04	0	0	301.21
CMIS3000-1000	26671242	0	0	162.09	0	0	549.05
CMIS5000-50	114866	228	-71.45	3678.86	0	-145.40	3132.34
CMIS5000-200	1335681	0	-1296.50	3902.63	0	-1305.10	2827.41
CMIS5000-500	7312462	0	-504.65	3800.53	0	-417.90	3126.76
CMIS5000-1000	27311381	0	-98.70	3678.42	0	-101.70	3799.58
CMIS5000-2000	104128699	0	0	1254.36	0	-133.20	3842.59

Table 5
Computational results of MWTS and ITS for CMC benchmark

Instances	BKR	MWTS			ITS		
		Gap_{bst}	Gap_{avg}	time	Gap_{bst}	Gap_{avg}	time
CMC1000-20	989	0	0	15.87	0	0	27.12
CMC1000-50	1179	0	0	26.67	0	0	39.18
CMC1000-100	1179	0	0	50.04	0	0	50.17
CMC1000-200	1179	0	0	58.58	0	0	144.47
CMC1000-400	1179	0	0	87.10	0	0	688.29
CMC3000-50	1411	0	-25.70	1800.50	0	-60.95	1637.05
CMC3000-100	1482	0	-89.40	2147.64	0	-113.45	1856.96
CMC3000-200	1417	65	-39.60	2096.12	0	-48.40	1519.60
CMC3000-500	1482	0	-130.90	1966.93	0	-155.45	1722.02
CMC3000-1000	1399	14	-59.50	1811.22	0	-99.25	1506.18
CMC5000-50	1641	0	-118.80	3486.37	0	-162.95	3832.24
CMC5000-200	1597	46	-94.90	3711.54	0	-111.00	4065.25
CMC5000-500	1618	22	-179.60	4149.87	0	-196.75	3319.48
CMC5000-1000	1711	-85	-264.60	4454.60	0	-270.85	4121.26
CMC5000-2000	1490	28	-89.15	3318.59	0	-114.45	3370.62

5 Analysis

This section presents systematic analyses of key parameters and algorithmic components in the MWTS algorithm. We first describe the details of parameter turning and conduct sensitivity analyses to see if there exists statistical differences among different parameter settings. Then, to shed light on the effects of various components, we compare the performances of the MWTS algorithm with its variants whose essential ingredients are modified.

5.1 Parameter analysis

The MWTS algorithm utilizes two kinds of parameters. The general control related parameters include: two thresholds of non-improvement iterations for restarting the search process and terminating the tabu search respectively (ns and nt), the tabu tenure (t) and the size of candidate list in the hybrid perturbation phase (clp), which use the recommended settings of Wang and Punnen (2017). The multi-wave related parameters include: the number of cycles for the forward and reverse phases during a wave (hn), the size of candidate list in the reverse phase (clh) and the threshold for fixing a vertex in the solution (ρ), whose recommended settings are determined by experiments with partial instance sets.

We select 10 representative instances for parameter tuning. Inspired by Glover (2016) and Song et al. (2019), the ranges of parameters hn, clh and ρ are set as $\{4, 5, 6, 7\}$, $\{2, 3, 4, 5\}$ and $\{4, 5, 6, 7\}$ respectively. For each parameter, we change its value within the range while keeping the other parameters unchanged, and perform the MWTS algorithm to solve each instance for 20 times. Then, by comparing the average result gap between BKR and the average objective value ($Gap_{avg} = (BKR - f_{avg})/BKR * 100$), we find that $hn = 5$, $clh = 2$ and $\rho = 5$ are the most appropriate setting in the current situation.

Table 6-8 report the average result gaps obtained by MWTS when parameters are tuned based on $hn = 5$, $clh = 2$ and $\rho = 5$, which show that this parameter setting can achieve better results for 10 instances, and is significantly better than other settings according to two-tailed sign test ($10 > CV^{15}_{0.05} \approx 8$). In addition, the Friedman test confirms that there exists statistical differences among different settings, and changing the value of hn, clh and ρ leads to significant differences with p-values of 0.0001, 0.0005 and 0.0001 respectively.

Table 6
Average result gaps under different hn (%)

Instances\\hn	4	5	6	7
cmc3000-50	2.1368	1.8427	4.0184	4.9398
cmc3000-100	6.7578	6.0729	6.8623	7.9993
cmc3000-200	5.6281	2.8229	3.3169	2.8934
cmc3000-500	10.6545	8.8394	10.1923	10.6511
cmc3000-1000	5.9507	4.2888	4.4174	5.8327
cmis3000-50	0.0393	0.0188	0.0885	0.0724
cmis3000-100	0.0187	0.0064	0.0123	0.0246
cmis3000-200	0.0017	0.0000	0.0002	0.0002
cmis3000-500	0.0001	0.0000	0.0001	0.0001
cmis3000-1000	0.0002	0.0000	0.0002	0.0002

Table 7
Average result gaps under different clh (%)

Instances\\clh	2	3	4	5
cmc3000-50	1.8427	2.4238	3.1999	3.1892
cmc3000-100	6.0729	7.6248	6.6700	7.8644
cmc3000-200	2.8229	3.3239	3.7368	3.4263
cmc3000-500	8.8394	10.4690	11.1808	10.2362
cmc3000-1000	4.2888	6.1115	6.6619	6.0186
cmis3000-50	0.0188	0.0420	0.1207	0.0769
cmis3000-100	0.0064	0.0150	0.0062	0.0123
cmis3000-200	0.0000	0.0032	0.0017	0.0002
cmis3000-500	0.0000	0.0001	0.0001	0.0001
cmis3000-1000	0.0000	0.0002	0.0002	0.0002

Table 8
Average result gaps under different ρ (%)

Instances\\ρ	4	5	6	7
cmc3000-50	3.5897	1.8427	4.8228	3.2140
cmc3000-100	8.8192	6.0729	7.9453	8.3063
cmc3000-200	3.8215	2.8229	5.1306	5.6351
cmc3000-500	11.7780	8.8394	11.6734	11.0290
cmc3000-1000	6.6405	4.2888	5.7505	7.6483
cmis3000-50	0.1878	0.0188	0.1207	0.1493
cmis3000-100	0.0372	0.0064	0.0383	0.0396
cmis3000-200	0.0032	0.0000	0.0032	0.0048
cmis3000-500	0.0001	0.0000	0.0001	0.0001
cmis3000-1000	0.0002	0.0000	0.0002	0.0002

5.2 Component analysis

In order to verify the effectiveness of the important ingredients in the proposed algorithm, we compare our MWTS algorithm with four variants, where the reverse moves are treated as the forward moves during AMR updating, the pure descent is applied in the forward phase instead of tabu search, the vertex fixation strategy and the hybrid perturbation phase are removed.

The additional experiments on 15 representative instances are carried out under the same experimental conditions as Sect. 4.1. We report the average result gaps between BKR and the average objective value ($Gap_{avg} = (BKR - f_{avg})/BKR * 100$) in Table 9, and analyze their performances in following sections.

Table 9
Average result gaps obtained by MWTS and its variants (%)

Instances	MWTS	MWTS-AMR	MWTS-DES	MWTS-NF	MWTS-NP
CMC3000-50	1.8426	2.2537	22.5584	2.6187	2.5372
CMC3000-100	6.0728	6.7510	26.4136	7.8171	6.8927
CMC3000-200	2.8228	3.0345	21.7819	3.6344	3.6062
CMC3000-500	8.8394	9.9055	25.9075	10.2834	9.4568
CMC3000-1000	4.2887	5.6754	22.6554	6.0686	5.6468
CMIS3000-50	0.0187	0.0464	2.3096	0.0563	0.0903
CMIS3000-100	0.0064	0.0042	1.6400	0.0064	0.0125
CMIS3000-200	0	0	0.9930	0.0016	0
CMIS3000-500	0	0	0.2948	0	0
CMIS3000-1000	0	0	0.0695	0	0
P4000.1	0.0026	0.0272	3.9761	0	0.0238
P4000.2	0.0100	0.0209	4.2489	0.0319	0.0124
P4000.3	0.0006	0.0127	3.9445	0.0195	0.0204
P4000.4	0.0224	0.0482	3.9715	0.0345	0.0703
P4000.5	0.0042	0.0220	4.1416	0.0040	0.0003

5.2.1 Effectiveness of AMR update strategy

During the forward phase, when a forward move happens to be another performed move' s reverse move, the algorithm deletes the record from AMR and ignores this reverse move, which leads to a decrease in the length of AMR. To assess its rationality, we produce a variant MWTS-AMR, where reverse moves are treated as normal forward moves and recorded in AMR, which makes the length of AMR unchanged. Table 9 shows that MWTS performs significantly better than the variant with 11 better, 3 equal and 1 worse results ($12 > CV^{15}_{0.05} \approx 11$). Further, Fig. 4 provides a group of box plots to compare the ranges and distributions of the average result gaps in Table 9, which shows that the dispersion degrees of two algorithms are similar, and the average results from MWTS are closer to BKR for CMC and Palubeckis instances, since the medians of MWTS are lower than the variant.

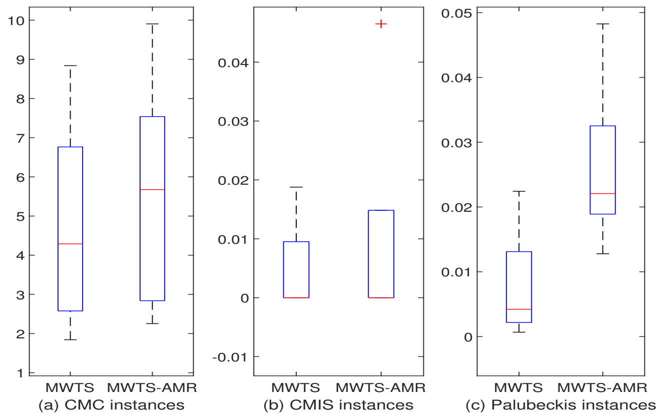


Fig. 4. Box plots of average result gaps obtained by MWTS and MWTS-AMR (%)

The observation confirms the effectiveness of the original AMR update strategy in MWTS. One possible reason for this result is that the AMR of the variant never decreases in any circumstances, which results in the excessive AMR growth and the early end of the search wave. Due to the short duration of the search wave, the search region may not be fully explored by the variant, which further causes the quality deterioration of the final solutions.

5.2.2 Effectiveness of the tabu search based forward phase

The MWTS algorithm employs tabu search for forward move selection in forward phase. To evaluate the merit of tabu search, we produce a variant MWTS-DES by replacing tabu search with pure descent as the main intensification component. Table 9 shows that MWTS obtains better average objective values than the variant for all 15 instances, and Fig. 5 indicates that without enhancements of tabu search, the average results of the variant become more unstable and far away from BKR.

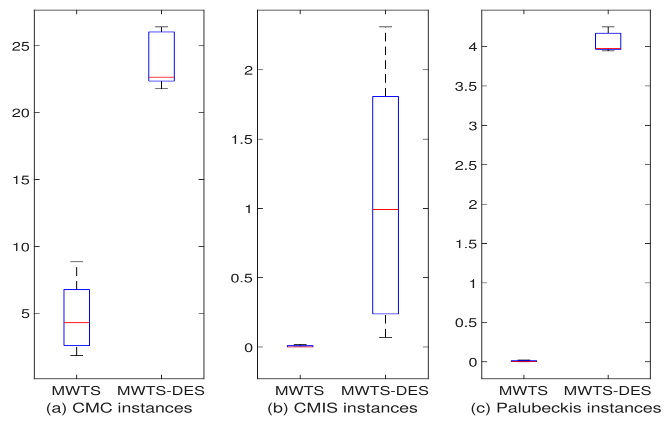


Fig. 5. Box plots of average result gaps obtained by MWTS and MWTS-DES (%)

This experiment concludes that the intensification of the MWTS algorithm is significantly enhanced by the tabu search. By prohibiting performed moves for a period of time, tabu search is able to avoid the trap of local optimum and achieve a strong search ability. Although the reverse phase can modify the current solution based on search information, the overall performance of the variant is not as good as the tabu search based MWTS algorithm, which further confirms the necessity of strengthening the forward phase.

5.2.3 Effectiveness of the vertex fixation strategy

During each search wave, the reverse phase implements the vertex fixation strategy to avoid those high value vertices from being dropped. To assess the impact of this strategy, we produce a variant MWTS-NF by removing this strategy from the MWTS algorithm while keeping other components unchanged. Table 9 indicates that MWTS achieves better performances than the variant with 10 better, 3 equal and 2 worse results ($11 = CV_{0.05}^{15} \approx 11$). In Fig. 6, the average results of MWTS do not vary much for CMIS and Palubeckis instances and have lower medians for all instance sets, which means that MWTS is more stable than the variant and its average results are closer to BKR generally.

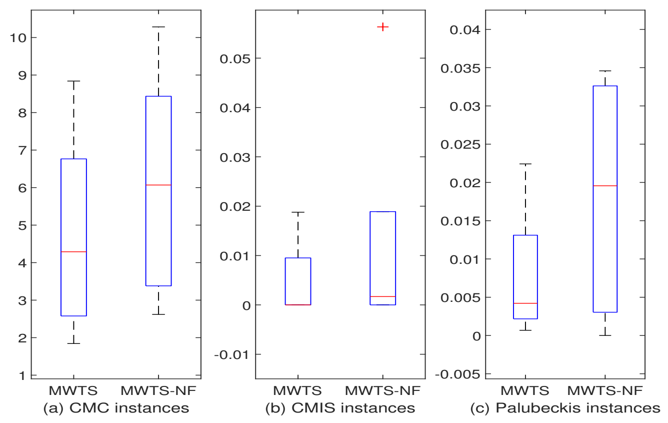


Fig. 6. Box plots of average result gaps obtained by MWTS and MWTS-NF (%)

The results disclose that the vertex fixation strategy is essential for the performance of the proposed algorithm. As more vertices are tried during the search process, the frequency information becomes progressively effective to evaluate the value of each vertex for the solution. Hence, it is worthwhile to utilize the information provided by vector Fre to inherit the good parts of the last solution. However, since the vector Fre based vertex fixation strategy has been removed, the variant can not guide the reverse phase based on frequency information, which causes the decline of the overall solution quality.

5.2.4 Effectiveness of the hybrid perturbation phase

The search diversification of our algorithm is enhanced by randomly performing the directed perturbation, the frequency guided perturbation and the recency guided perturbation. To assess the effectiveness of this approach, we design a variant MWTS-NP by removing the hybrid perturbation phase from the original algorithm. Table 9 shows that MWTS achieves 11 better, 3 equal and 1 worse results ($12 > CV^{15}_{0.05} \approx 11$), which is significantly better than the variant. Fig. 7 indicates that MWTS performs more stably than the variant for CMIS and Palubeckis instances, and obtains better average results for CMC and Palubeckis instances.

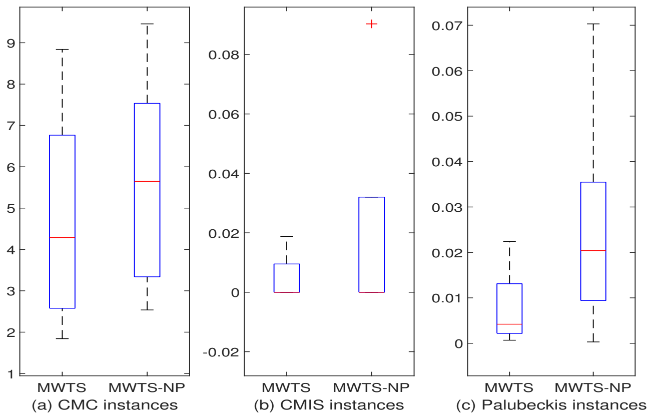


Fig. 7. Box plots of average result gaps obtained by MWTS and MWTS-NP (%)

This experiment demonstrates the necessity of the hybrid perturbation phase in the proposed algorithm. Different from Glover (2016), MWTS still keeps the tabu search running, when the forward phase encounters the interrupting point. In some respects, the reverse phase plays a similar role to the perturbation phase. However, the design purpose of the reverse phase is to adjust the solution based on the latest search information. The general performance of the variant confirms that once removing the perturbation component, the reverse phase alone can not guide the search process to a distant region.

6 Conclusion

In this study, we presented an effective multi-wave tabu search algorithm (MWTS) to solve the challenging boolean quadratic programming problem with generalized upper bound constraints. The proposed algorithm conducts waves to achieve search intensification. Each wave combines two alternating phases, a tabu search based forward phase that successively improves current solutions with AMR updating, and a reverse phase that adjusts current solutions according to AMR. The alternation repeats until reaching a local optimum. Between two waves, a hybrid perturbation phase that randomly chooses three different perturbation strategies is applied to ensure search diversification.

Computational experiments on four sets of benchmark instances indicate that the proposed algorithm is able to discover improved best solutions for 6 instances and match the previously best known solutions for most instances within competitive time. Furthermore, we conducted additional experiments to show the importance of the AMR update strategy, the advantage of the tabu search based forward phase, the necessity of the vertex fixation strategy and the hybrid perturbation phase.

Several important strategies applied in this work are general and could be applicable to solve other combinatorial optimization problems. For example, the strategy that guides the reverse and perturbation phases by frequency and recency information can be used in various metaheuristics. Moreover, the experimental evidences highlight the advantage of blending AMR and tabu list in the neighbourhood search for solving BQP-GUB, which inspire us to introduce automated tool, such as the conditional Markov Chain based method (Karapetyan et al., 2017), to evaluate the current algorithmic components, tune the key parameters and create effective metaheuristics.

Acknowledgements

We are grateful to the reviewers for their insightful and constructive comments, which helped us to significantly improvement the paper. This work was partially supported by the National Natural Science Foundation of China (No. 71971172), the Natural Science Basic Research Program of Shaanxi (No. 2020JM-089, No. 2020JQ-224, No. 2022JM-423), the Fundamental Research Funds for the Central Universities (No. 300102231656, No. D5000210834), Scientific Research Plan Project of Shaanxi Provincial Department of Education (No. 21JP007).

References

Anacleto, E. A., Meneses, C. N., and Liang, R. N. (2021). Fast r-flip move evaluations via closed-form formulae for boolean quadratic programming problems with generalized upper bound constraints. *Computers & Operations Research*, 132:105297.

Bookbinder, J. H. and Désilets, A. (1992). Transfer optimization in a transit network. *Transportation Science*, 26(2):106–118.

Bullnheimer, B. (1997). An examination scheduling model to maximize students' study time. In *International Conference on the Practice and Theory of Automated Timetabling*, pages 78–91. Springer.

Chang, J., Wang, L., Hao, J.-K., and Wang, Y. (2021). Parallel iterative solution-based tabu search for the obnoxious p-median problem. *Computers & Operations Research*, 127:105155.

Chen, M., Chen, Y., Du, Y., Wei, L., and Chen, Y. (2020). Heuristic algorithms based on deep reinforcement learning for quadratic unconstrained binary optimization. *Knowledge-Based Systems*, 207:106366.

Cook, D. J. and Holder, L. B. (2000). Graph-based data mining. *IEEE Intelligent Systems and Their Applications*, 15(2):32–41.

Daduna, J. R. and Voß, S. (1995). Practical experiences in schedule synchronization. In *Computer-Aided Transit Scheduling*, pages 39–55. Springer.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

Duffuaa, S. O. and Fedjki, C. A. (2012). General forms of the quadratic assignment problem. *International Journal of Operational Research*, 13(2):185–199.

Glasner, D., Vitaladevuni, S. N., and Basri, R. (2011). Contour-based joint clustering of multiple segmentations. In *CVPR 2011*, pages 2385–2392. IEEE.

Glover, F. (2000). Multi-start and strategic oscillation methods—principles to exploit adaptive memory. In *Computing Tools for Modeling, Optimization and Simulation*, pages 1–23. Springer.

Glover, F. (2016). Multi-wave algorithms for metaheuristic optimization. *Journal of Heuristics*, 22(3):331–358.

Glover, F., Hanafi, S., Guemri, O., and Crevits, I. (2018). A simple multi-wave algorithm for the uncapacitated facility location problem. *Frontiers of Engineering Management*, 5(4):451–465.

Glover, F., Laguna, M., and Marti, R. (2007). Principles of tabu search. *Approximation Algorithms and Metaheuristics*, 23:1–12.

Karapetyan, D., Punnen, A. P., and Parkes, A. J. (2017). Markov chain methods for the bipartite boolean quadratic programming problem. *European Journal of Operational Research*, 260(2):494–506.

Li, Z., Cheong, L.-F., Yang, S., and Toh, K.-C. (2018). Simultaneous clustering and model selection: Algorithm, theory and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(8):1964–1978.

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

Lü, Z. and Hao, J.-K. (2010). Adaptive tabu search for course timetabling. *European Journal of Operational Research*, 200(1):235–244.

Ma, F., Hao, J.-K., and Wang, Y. (2017). An effective iterated tabu search for the maximum bisection problem. *Computers & Operations Research*, 81:78–89.

Malucelli, F. (1996). A polynomially solvable class of quadratic semi-assignment problems. *European Journal of Operational Research*, 91(3):619–622.

Manohar, V., Vitaladevuni, S. N., Cao, H., Prasad, R., and Natarajan, P. (2011). Graph clustering-based ensemble method for handwritten text line segmentation. In *2011 International Conference on Document Analysis and Recognition*, pages 574–578. IEEE.

Pattillo, J., Youssef, N., and Butenko, S. (2012). Clique relaxation models in social network analysis. In *Handbook of Optimization in Complex Networks*, pages 143–162. Springer.

Raymond, J. W. and Willett, P. (2002). Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-aided Molecular Design*, 16(7):521–533.

Samorani, M., Wang, Y., Lv, Z., and Glover, F. (2019). Clustering-driven evolutionary algorithms: an application of path relinking to the quadratic unconstrained binary optimization problem. *Journal of Heuristics*, 25(4):629–642.

Shang, Z., Zhao, S., Hao, J.-K., Yang, X., and Ma, F. (2019). Multiple phase tabu search for bipartite boolean quadratic programming with partitioned variables. *Computers & Operations Research*, 102:141–149.

Skutella, M. (2001). Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM (JACM)*, 48(2):206–242.

Song, J., Wang, Y., Wang, H., Wu, Q., and Punnen, A. P. (2019). An effective multi-wave algorithm for solving the max-mean dispersion problem. *Journal of Heuristics*, 25(4):731–752.

Wang, Y., Lü, Z., Glover, F., and Hao, J.-K. (2012). Path relinking for unconstrained binary quadratic programming. *European Journal of Operational Research*, 223(3):595–604.

Wang, Y. and Punnen, A. P. (2017). The boolean quadratic programming problem with generalized upper bound constraints. *Computers & Operations Research*, 77:1–10.

Yamaguchi, A., Aoki, K. F., and Mamitsuka, H. (2004). Finding the maximum common subgraph of a partial k-tree and a graph with a polynomially bounded number of spanning trees. *Information Processing Letters*, 92(2):57–63.

Zhou, Y., Hao, J.-K., Fu, Z.-H., Wang, Z., and Lai, X. (2020). Variable population memetic search: A case study on the critical node problem. *IEEE Transactions on Evolutionary Computation*, 25(1):187–200.