Multiple phase tabu search for bipartite boolean quadratic programming with partitioned variables

Zhen Shang [a], Songzheng Zhao [a], Jin-Kao Hao [b,c], Xue Yang [d], Fuda Ma [b,*]

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

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

[c] *Institut Universitaire de France, 1 Rue Descartes, 75231 Paris, France*

[d] *School of Management, Xi'an Jiaotong University, Xianning West Road, 710049 Xi'an, China*

Computers & Operations Research, Oct. 2018

Abstract

The Bipartite Boolean Quadratic Programming Problem with Partitioned Variables (BBQP-PV) is an NP-hard problem with many practical applications. In this study, we present an effective multiple phase tabu search algorithm for solving BBQP-PV. The algorithm is characterized by a joint use of three key components: two tabu search phases that employ a simple neighborhood and a very large-scale neighborhood to achieve search intensification, and a hybrid perturbation phase that adaptively chooses a greedy perturbation or a recency-based perturbation for search diversification. Experimental assessment on 50 standard benchmarks indicates that the proposed algorithm is able to obtain improved lower bounds for 5 instances and match the previously best solutions for most instances, while achieving this performance within competitive time. Additional analysis confirms the importance of the innovative search components.

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

* Corresponding author.
Email addresses: 1007087859@qq.com (Zhen Shang), zhaosongzheng@nwpu.edu.cn (Songzheng Zhao), jin-kao.hao@univ-angers.fr (Jin-Kao Hao), yxueyx@outlook.com (Xue Yang), mada.nwpu@gmail.com (Fuda Ma).

1 Introduction

Let $G = (I, J, E)$ denote a bipartite graph with the two vertex sets $I = \{1, 2, \ldots, m\}$, $J = \{1, 2, \ldots, n\}$ and the edge set $E \subseteq I \times J$. The vertex set I is partitioned into p disjoint subsets $S_1, S_2, \ldots, S_p$ and the vertex set J is partitioned into k disjoint subsets $T_1, T_2, \ldots, T_k$. Further, each vertex $i \in I$ is associated with a weight c_i, each vertex $j \in J$ is associated with a weight d_j, and each edge $(i, j) \in E$ is associated with a weight q_{ij}. A subgraph $G' = (I', J', E')$ is said to be a representative subgraph of G, if the vertex set I' contains only one vertex, say α_r, of each disjoint subset $S_r, r = 1, 2, \ldots, p$, the vertex set J' contains only one vertex, say β_u, of each disjoint subset $T_u, u = 1, 2, \ldots, k$, and the edge set E' contains the edges connecting vertices between I' and J'. Then, the Binary Quadratic Programming Problem with Partitioned Variables (BBQP-PV) is to find such a representative subgraph G' that receives the maximum sum of edge weights and vertex weights. Formally, let $s = (s_\alpha, s_\beta)$ be a vertex set of G' where $s_\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_p\}$ and $s_\beta = \{\beta_1, \beta_2, \ldots, \beta_k\}$, BBQP-PV is to maximize the following objective:

$$\max f(s) = \sum_{r=1}^{p} \sum_{u=1}^{k} q_{\alpha_r, \beta_u} + \sum_{r=1}^{p} c_{\alpha_r} + \sum_{u=1}^{k} d_{\beta_u} \tag{1}$$

$$s.t.\ \alpha_r \in S_r, \text{for } r = 1, 2, \ldots, p \tag{2}$$

$$\beta_u \in T_u, \text{for } u = 1, 2, \ldots, k \tag{3}$$

Equivalently, BBQP-PV can be formulated as a constrained 0-1 quadratic program as follows [28].

$$\max f(x, y) = \sum_{i=1}^{m} \sum_{j=1}^{n} q_{ij} x_i y_j + \sum_{i=1}^{m} c_i x_i + \sum_{j=1}^{n} d_j y_j + c_0 \tag{4}$$

$$s.t. \sum_{i \in S_r} x_i = 1, for\ r = 1, 2, \ldots, p \tag{5}$$

$$\sum_{j \in T_u} y_j = 1, for\ u = 1, 2, \ldots, k \tag{6}$$

$$x_i, y_j \in \{0, 1\}, for\ i \in I, j \in J \tag{7}$$

where x_i (y_j) takes the value of 1 if $i = \alpha_r$ ($j = \beta_u$), otherwise the value of x_i (y_j) takes the value of 0. Without loss of generality, the constant c_0 is assumed to be 0.

BBQP-PV is a constrained version of the Bipartite Boolean Quadratic Pro-

gramming Problem (BBQP) [8,18,27]. It was recently proposed in [28] and proved to be strongly NP-hard. Moreover, they indicate that when $p = k = n$ and $|S_r| = |T_u| = n$ for any r and u, BBQP-PV turns out to be the Bipartite Quadratic Assignment Problem (BQAP) which in turn is a generalization of well-studied quadratic assignment problem (QAP).

BBQP-PV is a unified model of several classic combinatorial optimization problems, such as the Biclique Problem [3,14,25], the Max-induced Subgraph Problem [32,33], the Maximum Cut Problem on a Bipartite Graph [9,11,21] and the Matrix Factorization Problem [20,29,34]. Applications of the BBQP-PV model include clustering [4,30], location problem [7], social network analysis [17], bioinformatics [13,31] and many others.

Previous literature has reported many approaches for solving the closely related unconstrained BBQP problem. For example, [8] proposed an branch and bound algorithm and several iterated local search algorithms. [15] proposed multiple hybrid algorithms by combining tabu search and very large-scale neighborhood search strategies. [18] developed an effective Markov chain search algorithm. Moreover, bilinear programming algorithms are available for solving BBQP-PV [5,10,16] due to its bilinear objective function. However, without exploiting specific properties and structures of BBQP-PV, these general algorithms can not efficiently solve challenging BBQP-PV problem instances.

In the literature. The first computational study is proposed in [28], where several tailored local search and hybrid algorithms are developed and computational comparisons among the proposed algorithms are presented. Results show that the hybrid algorithms combining different neighborhoods outperform the algorithms that use these move operators in isolation and tabu search is a critical local search component. Another advanced metaheuristic algorithm recently proposed for solving BBQP-PV is an adaptive tabu search with strategic oscillation (ATS-SO) approach, which combines different move operators to collectively conduct neighborhood exploration and a history information guided strategic oscillation phase to diversify the search when the search gets trapped in local optimum. Computational assessments reveal that the ATS-SO algorithm outperforms the hybrid algorithms proposed in [28].

In this paper, we propose a new multiple phase tabu search (MPTS) algorithm for solving BBQP-PV. The proposed MPTS algorithm consists of a simple neighborhood based tabu search (SN-TS) phase and a very large-scale neighborhood based tabu search (VLSN-TS) phase for search intensification and a hybrid perturbation phase for search diversification [2,24,26]. The SN-TS phase aims to obtain a high-quality solution within a short period of time, while the VLSN-TS phase is dedicated to further refining the solution returned from the SN-TS phase. To escape local optimality and enable the search to

explore new promising regions, the hybrid perturbation phase is employed to build well-diversified solutions. According to the diversification requirement of the current search status, the perturbation phase adaptively chooses to use a greedy perturbation or a recency-based perturbation to improve diversification.

Evaluated on five sets of BBQP-PV benchmarks with a total of 50 instances, the proposed MPTS algorithm is able to find improved solutions for 5 instances and match the best known results for most instances within competitive computation time, performing better than recently proposed state-of-the-art algorithms in the literature. Additional analysis sheds light on the effectiveness of the incorporated components to the performance of the algorithm.

The rest of this paper is organized as follows. In Section 2, move operator definitions and fast evaluation methods are presented. Section 3 describes the main scheme and important components of the proposed algorithm. In Section 4, the computational results of our MPTS and the comparisons with state of the art algorithms in the literature are reported. Section 5 provides an experimental analysis of the key components used in the MPTS algorithm. Section 6 draws conclusions.

2 Move definitions and fast evaluation methods

In this section, we show two important properties of the BBQP-PV based on its graph representation. Then we present two types of moves along with their fast evaluation methods.

2.1 Problem properties

Based on the graph theoretical formulation of BBQP-PV given in Equations (1)-(3), we define, for any vertex $i \in I$ and $j \in J$,

$$C_i(s_\beta) = \sum_{u=1}^{k} q_{i,\beta_u} + c_i, i = 1, 2, \ldots, m \tag{8}$$

$$C_j(s_\alpha) = \sum_{r=1}^{p} q_{\alpha_r,j} + d_j, j = 1, 2, \ldots, n \tag{9}$$

$C_i(s_\beta)$ is called the objective contribution of the vertex i to the solution $s_\beta = (\beta_1, \beta_2, \ldots, \beta_k)$ when setting $\alpha_r = i(i \in S_r)$ and $C_j(s_\alpha)$ is called the objective

contribution of the vertex j to the solution $s_\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_p)$ when setting $\beta_u = j(j \in T_u)$.

Hence, the objective value $f(s)$ can also be expressed in terms of the objective contribution $C_i(s_\beta)$ and $C_j(s_\alpha)$ as:

$$f(s) = \sum_{r=1}^{p} C_{\alpha_r}(s_\beta) + \sum_{u=1}^{k} d_{\beta_u} = \sum_{u=1}^{k} C_{\beta_u}(s_\alpha) + \sum_{r=1}^{p} c_{\alpha_r} \tag{10}$$

Furthermore, we obtain the following two properties of BBQP-PV which have also been used for designing the very large scale neighborhood in [15,18,28].

Property 1: When s_β is fixed, the optimal $s^*_\alpha = (\alpha^*_1, \alpha^*_2, \ldots, \alpha^*_p)$ satisfies

$$C_{\alpha^*_r}(s_\beta) = \max_{i \in S_r} C_i(s_\beta), r = 1, 2, \ldots, p \tag{11}$$

Property 2: When s_α is fixed, the optimal $s^*_\beta = (\beta^*_1, \beta^*_2, \ldots, \beta^*_k)$ satisfies

$$C_{\beta^*_u}(s_\alpha) = \max_{j \in T_u} C_j(s_\alpha), u = 1, 2, \ldots, k \tag{12}$$

2.2 Swap moves

The SN-TS phase employs a simple swap move to perform neighborhood exploration. The swap move consists in replacing a vertex selected in the current solution with any vertex in the same subset, which generates a total of $m + n - p - k$ feasible neighbor solutions. The formal definition of the swap move is given as follows.

swap move: Given a solution $s = (s_\alpha, s_\beta)$, the swap move chooses a solution component α_r and replaces it with another vertex $i \in S_r \backslash \{\alpha_r\}$ to transform s to its neighbor solution $s' = (s'_\alpha, s_\beta)$ or chooses a solution component β_u and replaces it with another vertex $j \in T_u \backslash \{\beta_u\}$ to transform s to its neighbor solution $s' = (s_\alpha, s'_\beta)$.

Given that the whole search process generally performs the swap move for millions of iterations, it's essential to be able to quickly evaluate the objective gain of each swap move in each iteration. Based on the properties in Section 2.1, we employ a streamlined fast evaluation technique as follows. When a swap move is performed on s, the objective value of the resulting solution s' can be calculated by the following two equations. We apply Equation (13) if

α_r is changed to $\alpha'_r = i$ ($i \in S_r \backslash \{\alpha_r\}$). We apply Equation (14) if β_u is changed to $\beta'_u = j$ ($j \in T_u \backslash \{\beta_u\}$).

$$f(s') = f(s) - C_{\alpha_r}(s_\beta) + C_{\alpha'_r}(s_\beta) \tag{13}$$

$$f(s') = f(s) - C_{\beta_u}(s_\alpha) + C_{\beta'_u}(s_\alpha) \tag{14}$$

Since the current solution s is changed to s', $C_i(s_\beta)$ and $C_j(s_\alpha)$ also need to be updated. If a swap move is performed on β_u, we use Equation (15) to calculate $C_i(s'_\beta)$ and keep $C_j(s_\alpha)$ unchanged. In the same token, if a swap move is performed on α_r, we use Equation (16) to calculate $C_j(s'_\alpha)$ and keep $C_i(s_\beta)$ unchanged.

$$C_i(s'_\beta) = C_i(s_\beta) - q_{i,\beta_u} + q_{i,\beta'_u} \tag{15}$$

$$C_j(s'_\alpha) = C_j(s_\alpha) - q_{\alpha_r,j} + q_{\alpha'_r,j} \tag{16}$$

2.3 Very large-scale neighborhood moves

Many researchers have pointed out that the very large-scale neighborhood moves can usually reach better local optimal solutions [2] and algorithms using large neighborhoods have been successfully applied to solve a number of NP-hard problems [1,22]. Our algorithm adopts two very large-scale neighborhood moves, which were also used in [15,18,28].

α-optimal move: Given a solution s, choose a solution component α_r and replace it with another vertex $i \in S_r \backslash \{\alpha_r\}$. Based on the new s'_α, determine the optimal s'_β according to Property 2 to transform s to the neighbor solution s'.

β-optimal move: Given a solution s, choose a solution component β_u and replace it with another vertex $j \in T_u \backslash \{\beta_u\}$. Based on the new s'_β, determine the optimal s'_α according to Property 1 to transform s to the neighbor solution s'.

Specifically, we use the α-optimal move as an example to illustrate the details and complexity of such a move. To execute an α-optimal move, we first change the value of any α_r of s_α to obtain s'_α, which is actually a swap move on s_α. Then based on s'_α, we update all the $C_j(s_\alpha)$ to $C_j(s'_\alpha)$ using Equation (16). By following Property 2 and $C_j(s'_\alpha)$, we update s_β to the optimal s'_β. Finally, an

α-optimal move is performed. Furthermore, the time complexity of identifying the best optimal move is $O(nm)$ which is much higher than finding the best swap move with the complexity of $O(n+m)$.

Given the high computational complexity of the α-optimal and β-optimal moves, it's significant to be able to quickly determine the objective gain of performing such moves. Below we use the α-optimal move as an example to show the streamlined fast evaluation method.

Equation (9) indicates that a vertex i can influence $C_j(s_\alpha)$ of the vertex j, if and only if $q_{i,j} \neq 0$. Define $VI_i = \{j | q_{i,j} \neq 0, 1 \leq j \leq n\}$ and $TI_i = \{T_u | T_u \cap VI_i \neq \emptyset, 1 \leq u \leq k\}$. When an α-optimal move changes α_r to $\alpha'_r = i$ $(i \in S_r \backslash \{\alpha_r\})$, the optimal s'_β and $f(s')$ can be obtained by the following steps.

(1) Update $C_j(s_\alpha)$ of each vertex $j \in VI_{\alpha_r} \cup VI_{\alpha'_r}$ to $C_j(s'_\alpha)$ using Equation (16).
(2) For each T_u, if $T_u \in TI_{\alpha_r} \cup TI_{\alpha'_r}$, then β'_u is updated to be the vertex $j \in T_u$ whose $C_j(s'_\alpha)$ is maximum; otherwise $\beta'_u = \beta_u$. In this way, the optimal s'_β is obtained.
(3) Update $C_i(s_\beta)$ of each vertex $i \in I$ to $C_i(s'_\beta)$ using Equation (15).
(4) Calculate $f(s')$ using Equation (10).

By following a similar procedure, the objective function value of performing a β-optimal move can be efficiently calculated.

The BBQP-PV instances usually have many pairs of vertices i and j whose edge weight $q_{i,j} = 0$. It means that these types of vertices i and j are actually independent of each other. Thus we propose the above-mentioned method to remove unnecessary operations from the search procedure. However, it is obvious that for a BBQP-PV instance with many non-zero q_{ij}, this removal step will become helpless.

3 Multiple phase tabu search algorithm

In this section, we present the proposed multiple phase tabu search algorithm in details, including the main scheme, the initial solution generation, the simple neighborhood based tabu search phase (SN-TS), the very large-scale neighborhood based tabu search phase (VLSN-TS) and the perturbation phase.

3.1 Main scheme

Algorithm 1 presents the main scheme of our multiple phase tabu search algorithm. From a random initial solution, the MPTS algorithm repeats the following search rounds. For each round, it first executes a SN-TS phase that performs swap moves to obtain a high-quality solution (see Sect. 3.3). Then, a VLSN-TS phase is performed that executes the α-optimal move and the β-optimal move alternately to further refine the solution quality (see Sect. 3.4). After these search phases, a hybrid perturbation phase is triggered to produce a new diversified initial solution for the next round of tabu search. This perturbation phase adaptively applies a greedy perturbation or a recency-based perturbation depending on the current search status (see Sect. 3.5). The MPTS algorithm repeats the above-mentioned search rounds until a given stopping condition is satisfied.

Algorithm 1 Multiple phase tabu search algorithm

Input: A BBQP-PV instance.
Output: The best solution s^* found during the whole search procedure.
Set $s^* = \emptyset$, $f(s^*) = -\infty$, $gpnum = 0$.
Construct an initial solution s. (Sect. 3.2)
repeat
  $TL_i^{\alpha} = 0$, $TL_j^{\beta} = 0$, $i \in I$, $j \in J$. (Sect. 3.3)
  $Rec_i^{\alpha} = 0$, $Rec_j^{\beta} = 0$, $i \in I$, $j \in J$. (Sect. 3.5)
  $s \leftarrow$ SN-TS$(s, TL^{\alpha}, TL^{\beta}, Rec^{\alpha}, Rec^{\beta}, N1)$. (Sect. 3.3)
  if rand() mod $2 = 1$ **then**
    $s \leftarrow$ VLSN-TS: α-OptimalMove$(s, TL^{\alpha}, Rec^{\alpha}, N2)$. (Sect. 3.4)
  else
    $s \leftarrow$ VLSN-TS: β-OptimalMove$(s, TL^{\beta}, Rec^{\beta}, N2)$. (Sect. 3.4)
  end if
  if $f(s) > f(s^*)$ **then**
    $s^* = s$, $gpnum = 0$.
  else if $gpnum < gplimit$ **then**
    $s \leftarrow$ GreedyPerturbation(s, gps). (Sect. 3.5)
    $gpnum = gpnum + 1$.
  else
    $s \leftarrow$ RecencyPerturbation$(s, Rec^{\alpha}, Rec^{\beta}, rps)$. (Sect. 3.5)
    $gpnum = 0$.
  end if
until a stopping criterion is satisfied.

3.2 Initial solution

We use a two-step procedure to generate a feasible initial solution. First, each α_r $(1 \leq r \leq p)$ is assigned to a random vertex $i \in S_r$ to construct s_α. Then, based on property 2, the optimal s_β is obtained to complete the initial solution s. The complexity of this two-step initial solution generation procedure is bounded by $O(p+n)$. Preliminary experiments indicate the advantage of this strategy over the customary pure random strategy.

3.3 Simple neighborhood based tabu search phase

The SN-TS phase continually performs the swap moves to improve a given initial solution. This phase employs two different evaluation functions when selecting moves. The first evaluation function measures the exact objective gain, calculated as $-C_{\alpha_r}(s_\beta)+C_{\alpha'_r}(s_\beta)$ shown in Equation (13) or $-C_{\beta_u}(s_\alpha)+C_{\beta'_u}(s_\alpha)$ shown in Equation (14). The second evaluation function incorporates fluctuation in the objective gain and measures $C_{\alpha'_r}(s_\beta)$ or $C_{\beta'_u}(s_\alpha)$, representing the objective gain of setting $x_{\alpha'_r}=1$ while $x_{\alpha_r}=1$ or setting $x_{\beta'_u}=1$ while $x_{\beta_u}=1$. In the SN-TS phase, we use the first evaluation function with a probability of 0.9 and the second evaluation function with a probability of 0.1. All the moves in this phase are categorized as tabu moves and non-tabu moves depending on a tabu rule and an aspiration criterion [6,19,23].

The tabu rule requires that the reverse move of the performed move in each iteration is forbidden for the following tl iterations, where tl is called tabu tenure [12]. For this purpose, we use two lists TL^α and TL^β to record the iterations when the moves on s_α and s_β respectively are most recently performed. Assume that a move on s_α is composed of assigning α_r with the vertex i in place of the vertex j, then the tabu rule specifies $TL^\alpha_i = TL^\alpha_i + tl$ and $TL^\alpha_j = TL^\alpha_j + tl$ to forbid α_r from being assigned to j again for the next tl iterations. As long as a performed move does not include i and j, we assign $TL^\alpha_i = TL^\alpha_i - 1$ and $TL^\alpha_j = TL^\alpha_j - 1$. Hence, a move is determined as non-tabu if at least one of TL^α_i and TL^α_j equals 0 and is determined as tabu otherwise. Moreover, an aspiration criterion is applied to override the tabu rule if performing a tabu move is able to produce a solution that is better than the best solution found so far.

The SN-TS phase works as follows. Starting with an initial solution, it repeatedly performs iterations until the best solution can not be improved for $N1$ consecutive iterations. Each iteration includes the following three steps: 1) identify the tabu moves and non-tabu moves, 2) select an evaluation function and measure the objective values of all the moves using the fast evaluation

technique, 3) perform the non-tabu move with the maximum objective value or the move that satisfies the aspiration criterion. When the SN-TS phase finishes, it returns the best solution s found so far and the updated tabu lists.

3.4 Very large-scale neighborhood based tabu search phase

In order to improve the solution obtained from the SN-TS phase, the VLSN-TS phase is followed that alternatively uses the α-optimal moves and the β-optimal moves to perform the search. We use two similar evaluation functions in a probabilistic way as in the SN-TS phase, where the first measures the exact objective gain of performing a α-optimal move or a β-optimal move and the second measures $C_{\alpha'_r}(s_\beta)$ when performing a α-optimal move or $C_{\beta'_u}(s_\alpha)$ when performing a β-optimal move. The working scheme of the VLSN-TS phase is similar to the SN-TS phase but differs in the following aspects. First, the initial solution of the VLSN-TS phase is the best solution obtained in the SN-TS phase. Hence, it's more challenging to obtain an improving solution in the VLSN-TS phase. Second, a new tabu rule is designed for the α-optimal move and the β-optimal move. Since the optimal s_β is fully dependent on the swap move on s_α for a α-optimal move, we only use TL^α as the tabu list and apply a tabu rule that forces the performed swap moves to be forbidden for the following tl iterations. The tabu rule for the β-optimal moves follows the same idea. It's noteworthy that the initial tabu lists TL^α and TL^β are inherited from the SN-TS phase. Third, the maximum consecutive non-improvement iterations $N2$ to terminate the VLSN-TS phase is set to be much smaller than $N1$ in the SN-TS phase, considering that the evaluation of each α-optimal move and β-optimal move is more expensive than that of a swap move. Further experimental analysis confirms the effectiveness of the VLSN-TS phase to enhance the performance of the MPTS algorithm (see Section 5.1).

3.5 Perturbation phase

The perturbation phase starts to execute when the VLSN-TS phase finishes, which adaptively selects a greedy perturbation strategy or a recency-based perturbation strategy according to the diversification required in the current search status.

According to preliminary experimental observations, using the α-optimal move in the perturbation phase generally produces better results than the swap move. Thus, the greedy perturbation strategy employs the α-optimal move on the best solution found by the VLSN-TS phase to transform the input solution to a new initial solution for the next round of tabu search. Specifically, the greedy perturbation phase continually performs gps greedy steps (gps is a

parameter called the greedy perturbation strength), where each step assigns $\alpha_r = argmax\ C_i(s_\beta)$, $i \in S_r \backslash \{\alpha_r\}$ for a randomly chosen subset S_r and determines the optimal s_β. This greedy perturbation strategy does not deteriorate the objective value of the best solution too much and thus is beneficial to keep the new initial solution with good quality.

When the best solution can not be improved for consecutive *gplimit* rounds, the current search is judged as falling into a deep local optimum and a strong diversification is required. For this purpose, the recency-based perturbation phase is triggered that chooses a least recent assigned vertex from each subset and assigns the chosen vertices to generate a new solution. Specifically, during the search phases, we use two vectors Rec^α and Rec^β to store the latest iterations when the vertex i is assigned to α_r and the vertex j is assigned to β_u, respectively. The recency-based perturbation phase continually performs *rps* steps (*rps* is a parameter called the recency-based perturbation strength), where each step randomly chooses a subset S_r and assigns the vertex $i \in S_r \backslash \{\alpha_r\}$ with the minimum Rec_i^α or randomly chooses a subset T_u and assigns a vertex $j \in T_u \backslash \{\beta_u\}$ with the minimum Rec_j^β.

4 Computational experiments

This section reports the computational results of our proposed MPTS algorithm and comparisons with the state-of-the-art algorithms in the literature. Specifically, we first describe the benchmark instances and experimental protocol used in assessing the MPTS algorithm. Then we perform the parameter sensitivity analysis to show the effectiveness of the chosen values. Finally, we present experimental results and comparisons in details.

4.1 Benchmark instances and experimental protocol

We use the five benchmark sets of BBQP-PV instances generated in [28] to assess our algorithm, where each benchmark set contains 5 medium instances with $m = 200, 400, 600, 800, 1000$ and $n = 1000$ and 5 large instances with $m = 1000, 2000, 3000, 4000, 5000$ and $n = 5000$. Based on a bipartite graph $G = (I, J, E)$ where w_{ij} is the weight of edge $(i, j) \in E$, c_i and d_j are weights of the vertices $i \in I$ and $j \in J$, respectively, the instances of each benchmark set present the following characteristics:

- Random instances: $w_{i,j}, c_i, d_j$ obey normal distribution $N(0, 100^2)$ and $q_{i,j} = w_{i,j}$.
- Max Biclique instances: If $(i, j) \in E$, then $w_{i,j}$ obeys normal distribution

$N(100, 100^2)$; otherwise $w_{i,j}$ is a large negative number. And $q_{i,j} = w_{i,j}$, $c_i = 0$, $d_j = 0$.

- Max Induced Subgraph instances: If $(i, j) \in E$, then $w_{i,j}$ obeys normal distribution $N(100, 100^2)$; otherwise $w_{i,j} = 0$. And $q_{i,j} = w_{i,j}$, $c_i = 0$, $d_j = 0$.
- Max Cut instances: If $(i, j) \in E$, then $w_{i,j}$ obeys normal distribution $N(100, 100^2)$ and $q_{i,j} = -2w_{i,j}$; otherwise $q_{i,j} = w_{i,j} = 0$. And $c_i = \sum_{j=1}^{n} w_{i,j}$, $d_j = \sum_{i=1}^{m} w_{i,j}$.
- Matrix Factorization instances: Define a random matrix $H = (h_{i,j})$ and each $h_{i,j}$ has a probability of 0.5 to be assigned value 0 or value 1. Then $q_{i,j} = 1 - 2h_{i,j}$ and $c_i = 0$, $d_j = 0$.

Our MPTS algorithm is coded in C++ and compiled by GNU g++ on a computer with an Intel Xeon E5440 2.83GHz processor and 8GB RAM. The stopping condition is set to be 5 minutes for each medium instance and 60 minutes for each large instance. Given the stochastic nature of the proposed algorithm, each instance is solved by 20 independent runs. Table 1 presents the parameter setting in the MPTS algorithm.

Table 1
Settings of parameters

Parameters	Section	Description	Values
tl	3.3	tabu tenure	22
$N1$	3.3	continuous non-improvement iterations to terminate the SN-TS phase	50000
$N2$	3.4	continuous non-improvement iterations to terminate the VLSN-TS phase	25
gps	3.5	greedy perturbation strength	0.6
rps	3.5	recency-based perturbation strength	0.5
gplimit	3.5	continuous non-improvement tabu search rounds to switch the perturbation strategies	10

4.2 Parameter sensitivity analysis

To confirm the effectiveness of the values in Table 1, we additionally perform a parameter sensitivity analysis. We choose 10 representative instances to perform the experiment. For the experiment, we change the value of a chosen parameter while keeping the settings of all the other parameters unchanged and run the MPTS algorithm to solve each of the 10 instances. For each parameter setting and each instance, the average objective value over 20 runs is recorded.

For the results of each parameter, we apply the Friedman test to see if there exist statistical differences among different parameter settings. A parameter is sensitive if significant differences are observed among different settings. Otherwise, this parameter is considered as insensitive. The returned p-values of varying the values of parameters $N1$, $N2$, *gps* and *rps* are 0.069, 0.483, 0.967 and 0.763, respectively, meaning that there are no significant differences among

the considered settings for these parameters. However, changing the parameters tl and $gplimit$ leads to significant differences with p-values of 6.411e-04 and 0.012, respectively.

Tables 2 and 3 further present the average objective values of using each setting for the parameters tl and $gplimit$. As can be seen from the tables, the setting of $tl = 22$ and $gplimit = 10$ can achieve the best results for 5 out of the 10 tested instances, superior to other settings.

Table 2
Sensitivity analysis of the parameter tl

Instances/tl	f_{avg}						
	12	17	22	27	32	37	42
rand1000×5000	131858	131869	132050	131171	131606	131067	131109
rand2000×5000	203184	204232	204398	204042	203811	203750	201915
biclique1000×5000	311344	311344	311344	311344	311344	311344	311344
biclique2000×5000	567896	568225	568914	568688	567785	567571	568890
bimaxcut1000×5000	816968	816968	816968	816968	816968	816968	816968
bimaxcut2000×5000	1264870	1264870	1264870	1264870	1264870	1264870	1264870
maxinduced1000×5000	118398	118468	119001	118708	117914	118398	117599
maxinduced2000×5000	184509	184509	184509	184509	184509	184509	184509
matrixfactor1000×5000	1238	1234	1238	1230	1222	1220	1232
matrixfactor2000×5000	1956	1960	1964	1950	1954	1938	1942

Table 3
Sensitivity analysis of the parameter $gplimit$

Instances/$gplimit$	f_{avg}						
	6	7	8	9	10	11	12
rand1000×5000	131789	131842	131993	131908	132050	131737	131842
rand2000×5000	204123	203897	204232	204019	204398	204155	203898
biclique1000×5000	310644	311344	310930	311344	311344	311344	311344
biclique2000×5000	568688	569350	569350	569350	568914	568890	569072
bimaxcut1000×5000	816968	816968	816968	816968	816968	816968	816968
bimaxcut2000×5000	1264870	1264870	1264870	1264870	1264870	1264870	1264870
maxinduced1000×5000	118468	118442	118398	118708	119001	118708	118118
maxinduced2000×5000	184509	184509	184509	184509	184509	184509	184509
matrixfactor1000×5000	1230	1230	1234	1234	1238	1230	1234
matrixfactor2000×5000	1950	1954	1956	1954	1964	1952	1958

Table 4
Comparative results on medium instances

Instances	BKR	STS-OSLS			ATS-SO			MPTS		
		Gap_{bst}	Gap_{avg}	*time*	Gap_{bst}	Gap_{avg}	*time*	Gap_{bst}	Gap_{avg}	*time*
rand200×1000	95587	0	0	82.5	0	0	8.1	0	0	1.1
rand400×1000	149408	0	0	30.8	0	0	14.4	0	0	11.4
rand600×1000	195354	0	0	79.5	0	0	83.3	0	0	93.4
rand800×1000	244826	0	0	182.5	0	0	38.8	0	0	33.5
rand1000×1000	285491	0	-1	216.4	0	0	48.1	0	0	72.5
biclique200×1000	256419	0	-14	273.5	0	0	31.1	0	0	12.6
biclique400×1000	486625	0	0	75.4	0	0	18.3	0	0	27.9
biclique600×1000	709340	0	0	10	0	0	28.8	0	0	6.0
biclique800×1000	934429	0	0	52.4	0	0	110.3	0	0	14.1
biclique1000×1000	1152607	0	-135	162.8	0	0	36.4	0	0	105.9
bimaxcut200×1000	255582	0	0	1.5	0	0	9.7	0	0	0.6
bimaxcut400×1000	387937	0	0	5.5	0	0	10.2	0	0	1.3
bimaxcut600×1000	501074	0	0	8.6	0	0	9.7	0	0	7.17
bimaxcut800×1000	628120	0	0	17.3	0	0	20.4	0	0	2.3
bimaxcut1000×1000	762194	0	0	19.1	0	0	21.1	0	0	2.8
maxinduced200×1000	80289	0	0	78.7	0	0	13.7	0	0	2.9
maxinduced400×1000	124363	0	0	21.1	0	0	8.1	0	0	10.6
maxinduced600×1000	164565	0	0	20.4	0	0	36.9	0	0	11.3
maxinduced800×1000	208349	0	0	9.9	0	0	11.7	0	0	3.8
maxinduced1000×1000	245263	0	0	17.2	0	0	16.1	0	0	5.3
matrixfactor200×1000	930	0	0	76.3	0	0	10	0	0	1.1
matrixfactor400×1000	1446	0	0	41.6	0	0	17.3	0	0	6.4
matrixfactor600×1000	1940	0	0	86.3	0	0	39	0	0	12.3
matrixfactor800×1000	2392	0	0	97.9	0	0	33.6	0	0	16.9
matrixfactor1000×1000	2828	0	-1	210.5	0	0	72.4	0	0	165.9

4.3 Computational comparisons

In this section, we present experimental comparisons of the proposed MPTS algorithm with the STS-OSLS algorithm in [28] and the ATS-SO algorithm in [35]. It's noteworthy that ATS-SO and STS-OSLS are the best BBQP-PV algorithms recently published in the literature. All the algorithms were run on the same computing platform under the same stopping condition. Tables 4 and 5 present the results of MPTS, STS-OSLS and ATS-SO on the 25 medium instances and 25 large instances, respectively. For each algorithm, we report the best gaps Gap_{bst} between the best objective value f_{bst} obtained by each algorithm and the best known result BKR, the average gaps Gap_{avg} between the average objective value f_{avg} obtained by each algorithm and the best known result BKR, and the average running time *time* in seconds to attain the best objective value.

Table 5
Comparative results on large instances

Instances	BKR	STS-OSLS			ATS-SO			MPTS		
		Gap_{bst}	Gap_{avg}	*time*	Gap_{bst}	Gap_{avg}	*time*	Gap_{bst}	Gap_{avg}	*time*
rand1000×5000	132830	-3851	-5094	2284.7	0	0	1135.2	0	-780	2007.5
rand2000×5000	205455	-2275	-4195	1830.3	0	-573	1727.7	0	-1057	1458.8
rand3000×5000	270961	-1797	-2906	2071.9	0	-1263	1752.7	157	-1226	2217.1
rand4000×5000	333074	-2971	-4847	2330.8	0	-2049	2030.8	-218	-3137	2386.6
rand5000×5000	392923	-4105	-6297	1854.4	0	-3021	2350.3	-44	-3835	2049.4
biclique1000×5000	311344	0	-1097	2452.6	0	0	336.6	0	0	1954.0
biclique2000×5000	569350	-3863	-6084	1771.8	0	-43	1470.3	0	-436	2005.6
biclique3000×5000	824453	-28	-1096	1385	0	-18	1806.5	0	-788	2158.3
biclique4000×5000	1073688	0	-1201	1939.2	0	-147	1681	0	-1188	1772.2
biclique5000×5000	1324514	0	-979	1657.9	0	-53	2066.3	0	-814	1783.7
bimaxcut1000×5000	816968	0	0	46.3	0	0	70.8	0	0	11.7
bimaxcut2000×5000	1264870	0	0	33.5	0	0	96.1	0	0	13.9
bimaxcut3000×5000	1835507	0	0	37.1	0	0	117.4	0	0	18.6
bimaxcut4000×5000	2144147	0	0	72.4	0	0	130.8	0	0	35.6
bimaxcut5000×5000	2621300	0	0	211.4	0	0	141.3	0	0	30.7
maxinduced1000×5000	119222	-2285	-3344	2188.6	0	0	391.2	0	-221	1796.4
maxinduced2000×5000	184509	0	-2835	2013.4	0	0	533.9	0	0	707.2
maxinduced3000×5000	243649	-761	-1565	1792.8	0	-478	1963.9	0	-1125	1883.7
maxinduced4000×5000	298445	-686	-1693	1936.2	0	-771	2018.6	0	-1748	1206.4
maxinduced5000×5000	352207	-1463	-2710	1582.1	0	-1360	1830.6	0	-2006	1856.8
matrixfactor1000×5000	1252	-44	-53	1543.4	0	-5	1667.3	0	-13	1471.0
matrixfactor2000×5000	1972	-20	-36	1526.9	0	-8	1657.5	10	-7	1693.5
matrixfactor3000×5000	2618	-18	-27	1867	0	-11	1736.7	10	-12	1839.4
matrixfactor4000×5000	3230	-26	-47	1772.3	0	-15	2110.8	10	-16	1887.7
matrixfactor5000×5000	3818	-30	-60	1982.6	0	-19	1811.8	8	-25	1811.9

From Table 4, we find that our MPTS algorithm performs as well as ATS-SO and STS-OSLS in terms of the best objective values. In terms of the average objective values, MPTS and ATS-SO are able to reach BKR in each run, performing better than STS-OSLS which fails to do so for 4 instances. The computational time of MPTS is comparable to that of ATS-SO and much shorter than that of STS-OSLS.

From Table 5, we observe that MPTS performs better than ATS-SO and STS-OSLS by finding improved lower bounds for 5 instances and matching best known lower bounds for all except 2 instances. In terms of the average objective values, both MPTS and ATS-SO outperform STS-OSLS, but ATS-SO generally performs better than MPTS.

5 Analysis

To shed light on the behavior of the proposed algorithm, we assess in this section its essential ingredients including the combined use of two tabu search phases, the inherited tabu list strategy, the use of two evaluation functions in the search phases and the perturbation mechanism. All additional experiments shown below are conducted on 20 challenging instances and use the same parameter settings and stopping condition as indicated in Section 4.1.

5.1 Effectiveness of the combined use of the SN-TS and VLSN-TS phases

Our MPTS algorithm integrates the SN-TS phase with the VLSN-TS phase to improve an initial solution. In order to confirm its merit, we remove the SN-TS phase and the VLSN-TS phase respectively, while keeping other components unchanged to produce two variants: MPTS-NO-SN and MPTS-NO-VLSN.

To perform the experiment, we run MPTS-NO-SN and MPTS-NO-VLSN on each tested instance and summarize in Figure 1 the average objective gap (in percent) obtained by each variant with respect to MPTS, i.e., $\frac{f_{avg}(Variant)-f_{avg}(MPTS)}{f_{avg}(MPTS)}$. Figure 1 shows that MPTS performs better than MPTS-NO-VLSN and much better than MPTS-NO-SN. Specifically, MPTS-NO-VLSN finds worse average objective values than MPTS for 19 out of 20 instances and obtains an average percent gap of -1.04% over the 20 instances. MPTS-NO-SN finds worse average objective values than MPTS for all the instances and obtains an average percent gap of -2.34% over the 20 instances. To conclude, this experiment confirms that the performance of the proposed MPTS algorithm is enhanced by integrating the SN-TS phase and the VLSN-TS phase for conducting the neighborhood search.

5.2 Effectiveness of the inherited tabu list

Recall that in our MPTS algorithm, the VLSN-TS phase inherits the tabu list from the SN-TS phase rather than to use an independent tabu list. To evaluate the role of this strategy, we produce a new variant MPTS-NO-ITL that does not share the tabu list between the SN-TS phase and the VLSN-TS phase.

Figure 2 shows the average objective gaps obtained by MPTS-NO-ITL with respect to MPTS for each tested instance. The results indicate that MPTS performs better than MPTS-NO-ITL in terms of the average objective value. To be specific, MPTS-NO-ITL reaches worse average objective values for all

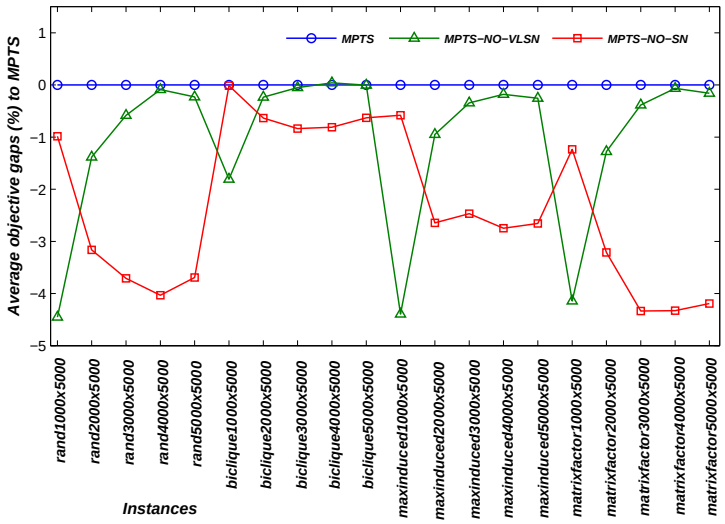

Fig. 1. Experimental comparisons among MPTS-NO-SN, MPTS-NO-VLSN and MPTS

instances and obtains an average gap of -0.58% over the 20 instances. This observation confirms that the search information recorded in the tabu list of the SN-TS phase is helpful to guide the subsequent VLSN-TS phase for reaching better solutions.

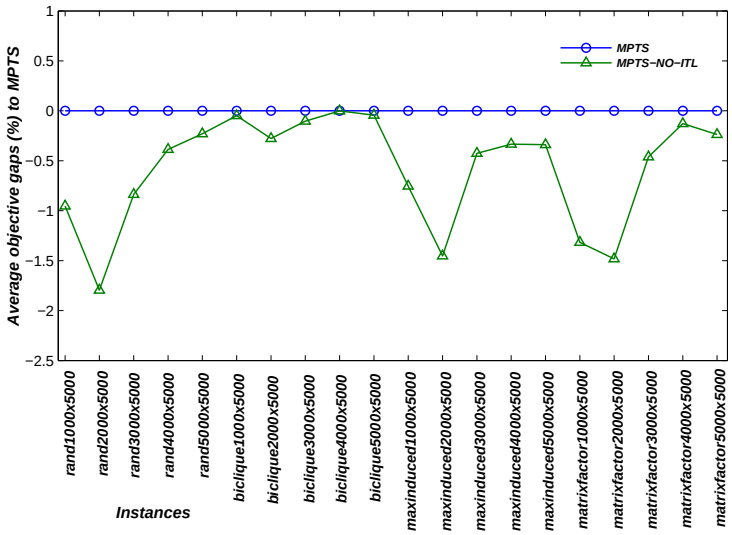

Fig. 2. Experimental comparisons between MPTS-NO-ITL and MPTS

5.3 Effectiveness of incorporating a second evaluation function

Our SN-TS and VLSN-TS search phases use two evaluation functions in a probabilistic way to evaluate moves. In order to show its effectiveness, we

produce a variant MPTS-NO-SEF that only keeps the customary objective function as the evaluation function in both search phases. For the tested instance, we show in Figure 3 the percent gaps of the average objective values obtained by MPTS-NO-SEF from those obtained by MPTS.

The results shows that MPTS-NO-SEF performs worse than MPTS for all instances and yields an average percent gaps of -0.47% over the 20 instances. Hence, this experiment demonstrates the merit of the incorporated additional evaluation function to the performance of the proposed MPTS algorithm.

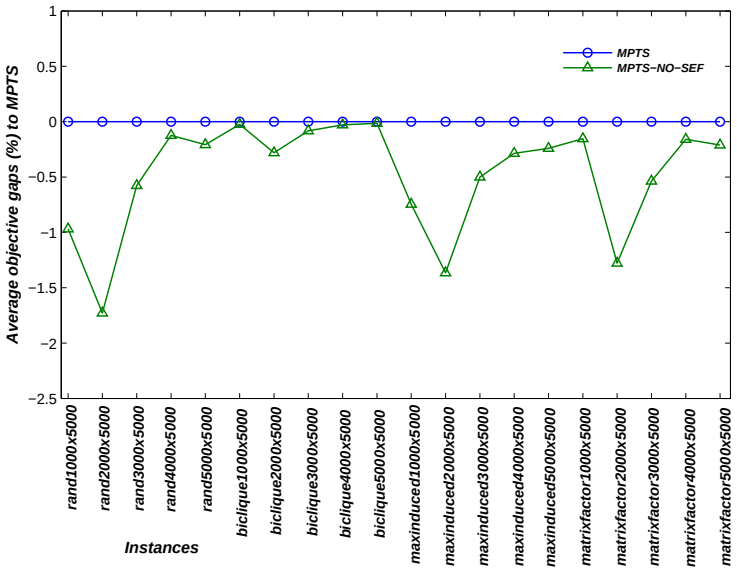

Fig. 3. Experimental comparisons between MPTS-NO-SEF and MPTS

5.4 Effectiveness of the proposed perturbation phase

The search diversification in the designed MPTS algorithm is achieved by adaptively selecting a greedy perturbation procedure or a recency-based perturbation procedure. To evaluate the impact of this hybrid perturbation strategy, we create two variants MPTS-NO-GP and MPTS-NO-RP by removing the greedy perturbation procedure and the recency-based perturbation procedure respectively. For each tested instance, the percent gaps of the average objective values of each variant to the average objective values of MPTS are shown in Figure 4.

Figure 4 shows that MPTS finds better average objective values than MPTS-NO-GP and MPTS-NO-RP for 19 instances and 20 instances respectively. Moreover, the percent gaps of the average objective values over 20 instances obtaind by MPTS-NO-GP and MPTS-NO-RP to those obtained by MPTS

Table 6
Results summary of MPTS and all the analyzed variants

Average	MPTS	MPTS-NO-SN	MPTS-NO-VLSN	MPTS-NO-ITL	MPTS-NO-SEF	MPTS-NO-GP	MPTS-NO-RP
f_{bst}	331553.8	326427.2	330163.2	330584.9	330713.6	330320.6	330762.0
f_{avg}	332472.3	328649.2	331211.1	331625.1	331715.3	331638.3	331728.4
time	1797.4	1785.3	1714.8	1787.1	1696.0	1762.2	1638.1

are -0.70% and -0.52%, respectively. To sum, this experiment demonstrates the superiority of the proposed hybrid perturbation strategy over the use of different perturbation strategies in isolation.

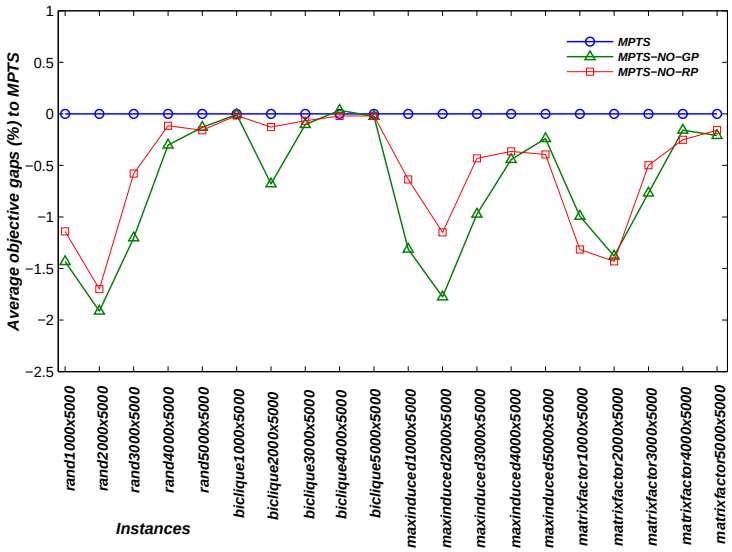

Fig. 4. Experimental comparisons among MPTS-NO-GP, MPTS-NO-RP and MPTS

5.5 Summary

Table 6 shows computational comparisons between MPTS and all the variants analyzed in previous experiments. For each algorithm, we summarize the averages of the best objective values f_{bst}, of the average objective values f_{avg} and of the computation time *time* over the 20 tested instances. From this table, we first observe that removing any component from our MPTS algorithm deteriorates the performance of the algorithm in terms of the best and average objective values. In addition, MPTS-NO-SN performs the worst, suggesting that removing the SN-TS phase is most destructive to the performance of our MPTS algorithm. Moreover, MPTS-NO-RP performs the best among all the variants, suggesting that removing the recency-based perturbation component has the least deterioration to the performance of the algorithm. To conclude, this summary provides insights into how different components contribute to the performance of the MPTS algorithm.

6 Conclusion

In this study, an effective multiple phase tabu search algorithm is developed to solve the challenging Bipartite Boolean Quadratic Programming Problem with Partitioned Variables. The proposed algorithm combines a simple neighborhood based tabu search phase with a very large-scale neighborhood based tabu search phase to achieve search intensification and employs a hybrid perturbation phase that adaptively selects a greedy perturbation or a recency-based perturbation to ensure search diversification. Extensive experiments indicate that the proposed algorithm is able to discover improved best solutions for 5 instances and match the previously best known solutions for most instances within competitive computation time.

Furthermore, we performed additional experiments to show the effectiveness of the chosen parameter settings, the importance of the inherited tabu list from the simple neighborhood based tabu search phase to guide the subsequent very large-scale neighborhood based tabu search phase for reaching high quality solutions, the merit of combining multiple tabu search phases,the advantage of incorporating a second evaluation function, as well as the role of the hybrid perturbation strategy.

Several important search strategies proposed in this work are general and could be applicable to solve other combinatorial optimization problems. For example, the hybrid perturbation strategy that adaptively applies a greedy perturbation or a recency-based perturbation depending on the current search status to locate starting solutions in promising search areas can be used as a general diversification component in various metaheuristics. When a solution is improved by multiple tabu search phases, an integrated tabu list could be more advantageous than several independent tabu lists. Furthermore, the strategy of integrating basic tabu search with very large-scale neighborhood based tabu search could be useful for a better search intensification.

Acknowledgment

The authors would like to thank the anonymous reviewers for their helpful comments and suggestions.

References

[1] Ahuja, R.K., Orlin, J.B., Sharma, D., 2001, Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem, *Mathematical Programming* , 91(1),71-97.

[2] Ahuja, R.K., Ergun, Ö., Orlin, J.B., Punnen, A.P., 2002, A survey of very large-scale neighborhood search techniques, *Discrete Applied Mathematics* , 123(1-3),75-102.

[3] Ames, B.P., Vavasis, S.A., 2011, Nuclear norm minimization for the planted clique and biclique problems, *Mathematical programming* , 129(1),69-89.

[4] Boros, E., Hammer, P.L., 1989, On clustering problems with connected optima in Euclidean spaces, *Discrete Mathematics* , 75(1-3),81-88.

[5] Bloemhof-Ruwaard, J.M. and Hendrix, E.M., 1996, Generalized bilinear programming: An application in farm management, *European Journal of Operational Research* , 90(1),102-114.

[6] Carrasco, R., Pham, A., Gallego, M., Gortázar, F., Martí, R., Duarte, A., 2015, Tabu search for the max-mean dispersion problem, *Knowledge-Based Systems* , 85,256-264.

[7] Dearing, P.M., Hammer, P.L., Simeone, B., 1992, Boolean and graph theoretic formulations of the simple plant location problem, *Transportation Science* , 26(2),138-148.

[8] Duarte, A., Laguna, M., Martí, R., Sánchez-Oro, J., 2014, Optimization procedures for the bipartite unconstrained 0-1 quadratic programming problem, *Computers & Operations Research* , 51,123-129.

[9] Festa, P., Pardalos, P.M., Resende, M.G., Ribeiro, C.C., 2002, Randomized heuristics for the MAX-CUT problem, *Optimization methods and software*, 17(6),1033-1058.

[10] Gallo, G., Ülkücü, A., 1977, Bilinear programming: an exact algorithm, *Mathematical Programming* , 12(1),173-194.

[11] Grötschel, M., Pulleyblank, W.R., 1981, Weakly bipartite graphs and the max-cut problem, *Operations Research Letters* , 1(1),23-27.

[12] Glover F., Laguna. M., 1997, Tabu search. *Kluwer Academic Publishers*, Boston.

[13] Gupta, Rohit., Rao, Navneet., Kumar, Vipin., 2011, Discovery of error-tolerant biclusters from noisy gene expression data, *BMC bioinformatics* , 12 Suppl 12. S1. 10.1186/1471-2105-12-S12-S1.

[14] Gillis, N., Glineur, F., 2014, A continuous characterization of the maximum-edge biclique problem, *Journal of Global Optimization* , 58(3),439-464.

[15] Glover, F., Ye, T., Punnen, A.P., Kochenberger, G., 2015, Integrating tabu search and VLSN search to develop enhanced algorithms: A case study using bipartite boolean quadratic programs, *European Journal of Operational Research* , 241(3),697-707.

[16] Jorge, J.M., 2005, A bilinear algorithm for optimizing a linear function over the efficient set of a multiple objective linear programming problem, *Journal of Global Optimization* , 31(1),1-16.

[17] Kochenberger, G.A., Hao, J.K., Lü, Z., Wang, H., Glover, F., 2013, Solving large scale max cut problems via tabu search, *Journal of Heuristics* , 19(4),565-571.

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

[19] Lai, X., Hao, J.K., 2016, A tabu search based memetic algorithm for the max-mean dispersion problem, *Computers & Operations Research* , 72,118-127.

[20] Miller, H.D., 1962, A matrix factorization problem in the theory of random variables defined on a finite Markov chain, *Mathematical Proceedings of the Cambridge Philosophical Society* , Vol. 58, No. 2,268-285, Cambridge University Press.

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

[22] Mouthuy, S., Van Hentenryck, P., Deville, Y., 2012, Constraint-based very large-scale neighborhood search, *Constraints* , 17(2),87-122.

[23] Ma, F., Hao, J.K., Wang, Y., 2017, An effective iterated tabu search for the maximum bisection problem, *Computers & Operations Research* , 81,78-89.

[24] Ma, F., Wang, Y., Hao, J.K., 2017, Path relinking for the vertex separator problem, *Expert Systems with Applications* , 82, 332-343.

[25] Peeters, R., 2003, The maximum edge biclique problem is NP-complete, *Discrete Applied Mathematics*, 131(3), 651-654,

[26] Pisinger, D., Ropke, S., 2010, Large neighborhood search, *Handbook of Metaheuristics*, 399-419.

[27] Punnen, A.P., Sripratak, P., Karapetyan, D., 2015, The bipartite unconstrained 0-1 quadratic programming problem: Polynomially solvable cases, *Discrete Applied Mathematics*, 193, 1-10.

[28] Punnen, A.P., Wang, Y., 2016, The bipartite quadratic assignment problem and extensions, *European Journal of Operational Research* , 250(3), 715-725.

[29] Strintzis, M., 1972, A solution to the matrix factorization problem, *IEEE Transactions on Information Theory*, 18(2), 225-232.

[30] Shen, B.H., Ji, S., Ye, J., 2009, Mining discrete patterns via binary matrix factorization, *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* , 757-766.

[31] Tanay, A., Sharan, R., Shamir, R., 2002, Discovering statistically significant biclusters in gene expression data, *Bioinformatics* , 18(suppl_1), S136-S144.

[32] Yang, X., Evans, D.J., Megson, G.M., 2005, Maximum induced subgraph of a recursive circulant, *Information Processing Letters*, 95(1), 293-298.

[33] Zvervich, I.E., Zverovich, V.E., 1995, An induced subgraph characterization of domination perfect graphs, *Journal of Graph Theory*, 20(3), 375-395.

[34] Zhu, F., Honeine, P., Kallas, M., 2014, Kernel nonnegative matrix factorization without the pre-image problem, *Machine Learning for Signal Processing (MLSP), 2014 IEEE International Workshop*, 1-6.

[35] Wang, Y., Wu, Q., Punnen, A. P., Glover, F., 2018, Adaptive tabu search with strategic oscillation for the bipartite boolean quadratic programming problem with partitioned variables, *Information Sciences*, 450, 284-300.