Solving the maximum vertex weight clique problem via binary quadratic programming

Yang Wang · Jin-Kao Hao* · Fred Glover · Zhipeng Lü · Qinghua Wu

Abstract In recent years, the general binary quadratic programming (BQP) model has been widely applied to solve a number of combinatorial optimization problems. In this paper, we recast the maximum vertex weight clique problem (MVWCP) into this model which is then solved by a Probabilistic Tabu Search algorithm designed for the BQP. Experimental results on 80 challenging DIMACS-W and 40 BHOSLIB-W benchmark instances demonstrate that this general approach is viable for solving the MVWCP problem.

Keywords: Maximum Vertex Weight Clique; Binary Quadratic Programming; Probabilistic Tabu Search.

In memory of Professor Wenqi Huang for his pioneer work on nature-inspired optimization methods

---

Yang Wang
School of Management, Northwestern Polytechnical University, 127 Youyi West Road, 710072 Xi'an, China
E-mail: yangw@nwpu.edu.cn

Jin-Kao Hao *(Corresponding author)*
1) LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France
2) Institut Universitaire de France, Paris, France
E-mail: jin-kao.hao@univ-angers.fr

Fred Glover
OptTek Systems, Inc., 2241 17th Street Boulder, CO 80302, USA
E-mail: glover@opttek.com

Zhipeng Lü
School of Computer Science and Technology, Huazhong University of Science and Technology, 430074 Wuhan, China
E-mail: zhipeng.lui@gmail.com

Qinghua Wu
School of Management, Huazhong University of Science and Technology, 430074 Wuhan, China
E-mail: qinghuawu1005@gmail.com

1 Introduction

Given an undirected graph $G = (V, E)$ with vertex set V and edge set E, a clique is a set of vertices $C \subseteq V$ such that every pair of distinct vertices of C is connected with an edge in G, i.e., the subgraph generated by C is complete. The maximum clique problem (MCP) is to find a clique of maximum cardinality. An important generalization of the MCP, known as the maximum vertex weight clique problem (MVWCP), arises when each vertex i in G is associated with a positive weight w_i. The MVWCP aims to find a clique of G with the maximum $\sum_{i \in C} w_i$. It is clear that the MCP is a special case of the MVWCP with $w_i = 1$ for each vertex.

The MCP is computationally difficult given that its associated decision problem is known to be NP-complete [10]. As a generalization of the MCP, the MVWCP has at least as the same computational complexity as the MCP. Like the MCP, the MVWCP has important applications in many domains like computer vision, pattern recognition and robotics [4].

To solve these clique problems, a variety of solution algorithms have been reported in the literature. Examples of exact methods based on the general Branch-and-Bound (B&B) or Branch-and-Cut methods for the MCP (or its equivalent maximum stable set problem) can be found in [8,22,23,25,28,32–34,36]. For the MVWCP, some exact algorithms are tightly related to the corresponding algorithms designed for the MCP [3,27] while other B&B based methods can be found in [38]. On the other hand, a number of heuristic algorithm have also been proposed to find sub-optimal solutions to the MVWCP, including an augmentation algorithm [26], a distributed computational network algorithm [6], a trust region technique algorithm [7], a phased local search algorithm [31], a multi-neighborhood tabu search algorithm [39], and a breakout local search algorithm [5]. For an updated recent review of algorithms for these clique problems, the reader is referred to [41].

During the past decade, binary quadratic programming (BQP) has emerged as a unified model for numerous combinatorial optimization problems, such as max-cut [20,37], set partitioning [24], set packing [2], generalized independent set [19] and maximum edge weight clique [1]. A review of the additional applications and the reformulation procedures can be found in [18,21]. Using the BQP model to solve the targeted problem has the advantage of directly applying an algorithm designed for the BQP rather than resorting to a specialized solution method. Moreover, this approach proves to be competitive for several problems compared to specifically designed algorithms [1,20,24,37].

There exists several studies on the application of the BQP model to solve the classic MCP [21,29,30]. However, for the more general MVWCP, no computational study has been reported in the literature using the BQP model. In this paper, we investigate for the first time the application of the BQP model to the MVWCP and solve the resulting BQP problem with the Probabilistic Tabu Search algorithm (BQP-PTS) designed for the BQP [37]. Experimental results on 80 challenging DIMACS-W and 40 BHOSLIB-W instances demonstrate that this general BQP approach with the PTS algorithm performs quite well in terms of solution quality at the price of more computing time for some benchmark instances.

The rest of this paper is organized as follows. Section 2 illustrates how to transform the MVWCP into the form of the BQP. Section 3 presents our Probabilistic Tabu Search algorithm to solve the transformed BQP model. Section 4 report the computational results and comparisons with other state-of-the-art algorithms in the literature. The paper concludes with Section 5.

2 Transformation to the BQP model

2.1 Linear model for the MVWCP

Given an undirected graph $G = (V, E)$ with vertex set V and edge set E, each vertex associated with a positive weight w_i, the binary linear programming model for the MVWCP can be formulated as follows [35]:

$$\begin{aligned} Max \quad & f(x) = \sum_{i=1}^{n} w_i x_i \\ \text{subject to:} \quad & x_i + x_j \leq 1,\ \forall \{v_i, v_j\} \in \overline{E} \\ & x_i \in \{0,1\}, i \in \{1, \ldots, n\} \end{aligned} \tag{1}$$

where $n = |V|$, x_i is the binary variable associated to vertex v_i, $\overline{E}$ denotes the edge set of the complementary graph $\overline{G}$.

Notice that if $w_i = 1$ ($i \in \{1, \ldots, n\}$), Eq. (1) turns into the linear model of the classic maximum clique problem.

2.2 Nonlinear BQP alternative

The linear model of the MVWCP can be recast into the form of the BQP by utilizing the quadratic penalty function $g(x) = Px_ix_j$ (x_i is binary, $i \in \{1, \ldots, n\}$) to replace the inequality constraint of the MVWCP where P is a negative penalty scalar. Since the inequality constraint $x_i + x_j \leq 1$ implies that x_i and x_j cannot receive value 1 at the same time, the infeasibility penalty function $g(x)$ will equal to 0 if the inequality constraint is satisfied; otherwise $g(x)$ will take a large penalty value $2P$. To construct the nonlinear BQP model, each inequality constraint is replaced by the penalty function $g(x)$ which is added to the linear objective of Eq. (1) and the nonlinear BQP model can be formulated as follows:

$$\begin{aligned} Max \quad & xQx = \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} w_{ij} x_i x_j \\ & x_i \in \{0,1\}, i \in \{1, \ldots, n\} \end{aligned} \tag{2}$$

where $w_{ij} = P$ if $\{v_i, v_j\} \in \overline{E}$ and 0 otherwise.

This formulation is one of many nonlinear reformulations of the MVWCP and has been studied in previous work like [17]. The quadratic function will have the same objective value as the linear form subject to all penalty items equaling to 0, indicating that all constraints are satisfied. According to Eq. (2), any violated constraint, i.e., for each $\{v_i, v_j\} \in \overline{E}$, in a solution will add a penalty value $2P$ to the objective value. Thus, simply setting $|P| > 0.5\sum_{i=1}^{i} w_i$, where each linear objective function coefficient $w_i > 0$, will enable an infeasible solution to get a large penalty value. Actually it suffices to set a smaller $|P| > 0.5w^m$ (w^m is the maximal value among all $w_i,\ i \in \{1, \ldots, n\}$). Under this setting, a good decision for improving an infeasible solution would be to remove vertices associated with violated constraints, making constraints gradually reduced and finally an infeasible solution become feasible. Consider that the quadratic penalty function should be negative under the case of a maximal objective, we select $P = -1000$ for the MVWCP benchmark instances tested in our experiments. With this choice, for any optimal solution x of problem (2), $g(x) = 0$ holds. In other words, the subgraph constructed by the variables with the assignment of 1 in the optimized solution x forms a clique. An illustrative example of this transformation is given in Appendix. Since Eq. (2) corresponds to the well-known BQP model, any algorithm designed for solving the BQP can be readily used to solve the MVWCP. In our case, we apply a probabilistic tabu search algorithm described in the next section.

3 Probabilistic tabu search algorithm

Metaheuristics are often used to solve hard optimization problems, such as quasi-human based heuristics [16,40], variable neighborhood search [15], ant colony algorithm [9], probabilistic tabu search [11,42], etc. In this paper, we solve the MVWCP directly in the nonlinear BQP form as expressed in Eq. (2) by adapting our previous Probabilistic Tabu Search algorithm (BQP-PTS) designed for the BQP [37]. BQP-PTS is a multistart procedure, consisting of a greedy probabilistic solution construction phase and a sequel tabu search phase to optimize the objective function. These two phases proceed iteratively until a stopping condition is satisfied. Below we summarize the main ingredients of the BQP-PTS algorithm.

3.1 Greedy probabilistic construction of initial solutions

We construct a new solution for the general BQP model according to a greedy probabilistic construction heuristic. The construction procedure consists of two phases: one is to adaptively and iteratively select some variables to receive the value 1; the other is to assign the value 0 to the remaining variables. The pseudo-code of this construction procedure is shown in Algorithm 1.

First, the partial solution is set to be empty and all the variables of the problem instance are put into the set of the remaining variables VS. At each

Algorithm 1 Outline of the greedy probabilistic construction heuristic

Let px denote the partial solution and VS denote variables not in the partial solution, initialize $px = \emptyset$, $VS = \{x_1, x_2, \ldots, x_n\}$
repeat
  Construct a candidate list $CL \subset VS$ where each variable x_j in CL has a positive objective function increment OFI, calculated as $OFI_j = w_j + \sum_{x_i \in px} w_{ij}$
  Choose randomly one variable x_s from CL with a probability of $1/|CL|$ and set $x_s = 1$
  Enlarge the partial solution with $px = px \cup \{x_s\}$
  Update VS with $VS = VS \setminus \{x_s\}$
until $CL = \emptyset$
Set $x_i = 0$ for $\forall x_i \in VS$

iteration we construct a candidate list CL such that CL is a subset of VS and each variable in CL has a positive objective function increment OFI. Then we choose one variable from CL with a probability of $1/|CL|$ and assign it with the value 1. This variable with its assignment value is added into the partial solution and is removed from VS. This process continues until CL becomes empty. The last step is to assign the remaining variables in VS with value 0.

To quickly compute the objective function increment OFI, we maintain a vector IV, with each entry IV_i recording the objective function increment when putting a variable x_i with the value 1 into the partial solution. Initially, IV is computed as w_i since the initial partial solution is empty. Once a variable x_s joins into the partial solution, then each such entry IV_i with its corresponding variable belonging to the set of the remaining variables VS is updated as $IV_i = IV_i + 2w_{si}$. Because of this additional vector, the complexity of this construction procedure is bounded by $O(n)^2$.

Although this strategy is much simpler than that used in the original algorithm [37], it was experimentally demonstrated to be effective. Notice that seen from the side of the MVWCP, CL is the set of vertices which form a clique with those in the partial solution. This strategy of constructing an initial solution is consistent with many specific maximum clique algorithms in the literature.

3.2 Tabu search

For each initial solution generated by the greedy probabilistic construction, we apply an extended version of the tabu search algorithm described in [37] to further improve its quality. The tabu search algorithm in [37] uses a simple one-flip move (flipping the value of a single variable x_i to its complementary value $1 - x_i$) to conduct the neighborhood search. Here we not only exploit the one-flip move but also incorporate a two-flip move (flipping the values of a pair of variables (x_i, x_j) to their corresponding complementary values $(1 - x_i, 1 - x_j)$) [13]. The above two types of moves constitute the neighborhood structures N1 and N2.

One drawback of an N2 move is the amount of time it consumes. Considerable effort is required to evaluate all the two-flip moves because the neighborhood structure N2 generates $n(n-1)/2$ solutions at each iteration. To overcome this obstacle, we employ an instance of the Successive Filter candidate list strategy of [14] by restricting our attention to moves in N2 that can be obtained as follows. The first step is to examine all the one-flip moves of the current solution x, and if any of these moves is improving we go ahead and select it. But if no one-flip move is improving, we limit attention to one-flip moves that produce an objective function value no worse than $f(x) + 2P$, where $f(x)$ is the objective function value of x. (Recall that we are maximizing and the penalty P is negative. This implies that the candidate one-flip moves can violate at most a single additional constraint beyond those violated by x, since a single constraint is penalized as $Px_{ij} + Px_{ji}$ and hence incurs a penalty of $2P$.) Finally, only the one-flip moves that pass this filtering criterion are allowed to serve as the source of potential two-flip moves.

Tabu search typically introduces a tabu list to assure that solutions visited within a certain number of iterations, called the tabu tenure, will not be revisited [14]. In the present study, each time a variable x_i is flipped, this variable enters into the tabu list and cannot be flipped for the next $TabuTenure$ iterations. For the neighborhood structure N1, our tabu search algorithm then restricts consideration to variables not forbidden by the tabu list. For the neighborhood structure N2, we consider a move to be non-tabu only if both variables associated with this move are not in the tabu list and only such moves are considered during the search process. According to preliminary experiments, we set $TabuTenure(i) = 7 + rand(5)$ where rand(5) produces a random integer from 1 to 5.

For each iteration in our tabu search procedure, a non-tabu move is chosen according to the following rules: (1) if the best move from N1 leads to a solution better than the best solution obtained in this round of tabu search, we select the best move from N1, thus bypassing consideration of N2; (2) if no such move in N1 exists, we select the best move from the combined pool of N1 and N2. A simple aspiration criterion is applied that permits a move to be selected in spite of being tabu if it leads to a solution better than the current best solution. The tabu search procedure stops when the best solution cannot be improved within a given number μ of moves and we call this number the *improvement cutoff*. According to a preliminary experiment on parameter tuning, μ is set to be 5000 for all the instances except for san instances for which $\mu = 10$. In fact, it was observed that for some san instances, it is more effective to restart the search than to make long tabu iterations.

In order to quickly calculate the gains of performing a move, we maintain a vector Δ which is initialized as follows:

$$\Delta_i = \begin{cases} w_i + \sum_{j=1,j\neq i}^{n} 2w_{ij}x_j & (x_i = 0) \\ w_i - \sum_{j=1,j\neq i}^{n} 2w_{ij}x_j & (x_i = 1) \end{cases} \quad (3)$$

Then if a move corresponding to a one-flip move x_i is performed, then we update the set of variables affected by this move using the following scheme [12]:

$$\Delta_k = \begin{cases} -\Delta_k \quad (k = i) \\ \Delta_k - 2w_{ik} \quad (k \neq i, x_k = x_i) \\ \Delta_k + 2w_{ik} \quad (k \neq i, x_k = 1 - x_i) \end{cases} \tag{4}$$

If a move corresponding to a two-flip move (x_i, x_j) from the neighborhood N2 is performed, then we update the set of variables affected by this move using the following scheme [13]:

$$\Delta_k = \begin{cases} -\Delta_k - 2w_{ij} \quad (k = i \ or \ k = j) \\ \Delta_k - 2w_{ik} + 2w_{jk} \quad (k \neq i, k \neq j, x_k = x_i, x_k = 1 - x_j) \\ \Delta_k + 2w_{ik} - 2w_{jk} \quad (k \neq i, k \neq j, x_k = x_j, x_k = 1 - x_i) \end{cases} \tag{5}$$

Given the fact that the BQP-PTS algorithm is designed for the general BQP model (instead of the MVWCP model studied in the paper), the above presentation of BQP-PTS does not refer to the MVWCP. However, it is possible to give an interpretation of some operators used by BQP-PTS related to the MVWCP. For instance, the one-flip move for the BQP model can be considered as moving a single vertex in or out the current solution (clique). On the other hand, such an interpretation will change depending on the target problem under consideration.

4 Experimental results

4.1 Benchmark instances

We used two sets of benchmark instances for our computational assessments. The first set concerns 80 DIMACS-W instances proposed in [31], which were adapted from the well-known DIMACS instances[1] for benchmark purpose to evaluate maximum clique algorithms. The second set is composed of 40 BHOSLIB-W instances[2], which were adapted from the BHOSLIB benchmarks with hidden optimum solutions [5]. The weighting method is to allocate weights to vertices according to the following scheme: for each vertex i, w_i is set equal to $i \ mod \ 200 + 1$, which enables us to exactly replicate the instances without difficulty.

The DIMACS benchmarks comprise the following families of graphs: Random graphs (Cx.y and DSJCx.y of size x and density 0.y), Steiner triple graphs (MANNx with up to 3321 nodes and 5506380 edges), Brockington graphs with hidden optimal cliques (brockx_1, brockx_2, brockx_3, brockx_4 of size x), Gen random graphs with a unique known optimal solution (genx_p0.9_z of size x), Hamming and Johnson graphs stemming from the coding theory, Keller graphs based on Keller's conjecture on tilings using hypercubes (with up to

[1] http://cs.hb g.psu.edu/txn131/clique.html

[2] http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm

Algorithm 2 Outline of the tabu search algorithm

Input: a given solution x with its solution value $f(x)$
Output: the local optimal solution x^* with its solution value $f(x^*)$
TL: an n-dimensional vector for maintaining the tabu list Δ: an n-dimensional vector for recording the move gain of performing each one-flip move
Initialize Δ according to Eq. (3), $TL_i = 0$ for all $i = 1$ to n
Set $NonImp = 0$, $Iter = 0$
while $NonImp < \mu$ (μ is called improvement cutoff) **do**
  Identify the best non-tabu one-flip move or the best one-flip move that is tabu but satisfies the aspiration rule from the neighborhood N1, say this move corresponds to flipping x_i
  if $f(x) + \Delta_i > f(x^*)$ **then**
    $x_i = 1 - x_i$, $f(x) = f(x) + \Delta_i$
    Update Δ according to Eq. (4)
    Update Tabu List by setting $TL_i = Iter + TabuTenure_i$
  else
    Identify the best non-tabu move or the best tabu move that satisfies the aspiration rule from the neighborhood N1 and N2
    if this move corresponds to flipping x_i **then**
      $x_i = 1 - x_i$,$f(x) = f(x) + \Delta_i$
      Update Δ according to Eq. (4)
      Update Tabu List by setting $TL_i = Iter + TabuTenure_i$
    end if
    if this move corresponds to flipping x_i and x_j **then**
      $x_i = 1 - x_i$, $x_j = 1 - x_j$, $f(x) = f(x) + \Delta_i + \Delta_j + 2w_{ij}$
      Update Δ according to Eq. (5)
      Update Tabu List by setting $TL_i = Iter + TabuTenure_i$, $TL_j = Iter + TabuTenure_j$
    end if
  end if
  if $f(x) > f(x^*)$ **then**
    $x^* = x$, $f(x^*) = f(x)$, $NonImp = 0$
  else
    $NonImp = NonImp + 1$
  end if
  $Iter = Iter + 1$
end while

3361 verices and 4,619,898 edges), P-hat graphs (p_hatx-z of size x), San random graphs (sanx_y_z of size x and density 0.y) and Sanr random graphs (sanrx-z with size x and density z). The BHOSLIB-W benchmarks have sizes ranging from 450 vertices and 17,794 edges up to 1534 vertices and 127,011 edges.

4.2 Experimental protocol

Our Probabilistic Tabu Search algorithm for the BQP model was programmed in C++ and compiled using GNU GCC on a PC with Pentium 2.83GHz CPU and 2GB RAM. We used the CPU clocks as the stop condition of our algorithm. Given the stochastic nature of BQP-PTS, each problem instance was independently solved 100 times.

For the DIMACS-W benchmarks, the time limit for a single run was set as follows: 1 minute for instances of hamming, gen, c-fat, johnson, p_hat, sanr, keller except keller6 and mann_a9; 5 minutes for instances of brock, dsjc, san and C families except C2000.5, C2000.9, C4000.5; 60 minutes for C2000.5, C2000.9 and keller6; 600 minutes for C4000.5, mann_a27, mann_a45, mann_a81. For the BHOSLIB benchmarks, the time limit was set as 60 minutes.

4.3 Experimental results

In this section, we verify the effectiveness of our BQP approach with the BQP-PTS algorithm on the 80 DIMACS-W instances and 40 BHOSLIB-W instances. Furthermore, we compare this general BQP approach with three recent and powerful heuristics which are specially dedicated to the MVWCP: the PLS_W algorithm [31], the multi-neighborhood tabu search algorithm MS/TS [39] and the breakout local search BLS [5].

Table 1: Computational comparisons of the BQP-PTS approach with the PLS, MS/TS and BLS algorithms on the set of DIMACS-W instances

Instance	BQP-PTS			PLS_W [31]			MS/TS [39]			BLS [5]		
	Best	Succ.	Time	Best	Succ.	Time	Best	Succ.	Time	Best	Succ.	Time
brock200_1	2821	100	0.02	2821	100	0.19	2821	100	$< \epsilon$	2821	100	$< \epsilon$
brock200_2	1428	100	0.08	1428	100	0.02	1428	100	$< \epsilon$	1428	100	0.03
brock200_3	2062	100	0.09	2062	100	0.01	2062	100	$< \epsilon$	2062	100	0.01
brock200_4	2107	100	0.22	2107	100	0.70	2107	100	$< \epsilon$	2107	100	0.01
brock400_1	3422	100	0.72	3422	32	437.19	3422	32	0.03	3422	100	0.05
brock400_2	3350	100	1.00	3350	61	415.95	3350	61	0.03	3350	100	0.08
brock400_3	3471	100	0.57	3471	100	12.04	3471	100	0.03	3471	100	0.26
brock400_4	3626	100	4.01	3626	100	0.05	3626	100	4.70	3626	100	7.60
brock800_1	3121	100	3.95	3121	100	31.46	3121	100	0.05	3121	100	0.13
brock800_2	3043	100	42.29	3043	69	893.42	3043	100	0.20	3043	69	0.51
brock800_3	3076	100	8.22	3076	100	3.35	3076	100	0.08	3076	100	0.50
brock800_4	2971	8	105.53	2971	100	3.77	2971	100	49.70	2971	100	339.07
C125.9	2529	100	0.02	2529	100	8.08	2529	100	0.02	2529	100	0.01
C250.9	5092	100	0.05	5092	17	247.69	5092	100	0.06	5092	100	0.06
C500.9	6955	100	0.21	**6822**	–	–	6955	100	0.07	6955	100	0.25
C1000.9	9254	100	37.50	**8965**	5	344.74	9254	100	8.90	9254	100	12.33
C2000.5	2466	71	1366.51	2466	18	711.27	2466	100	1.84	2466	100	2.1
C2000.9	10999	72	2711.97	**10028**	–	–	10999	22	168.11	10999	74	1152.78
C4000.5	2792	19	19902.77	2792	–	–	2792	100	80.56	2792	100	179.89
DSJC500.5	1725	100	3.82	1725	100	0.95	1725	100	0.04	NA	NA	NA
DSJC1000.5	2186	81	115.42	2186	100	47.76	2186	100	0.20	NA	NA	NA
keller4	1153	100	0.05	1153	100	0.02	1153	100	0.03	1153	100	0.04
keller5	3317	100	5.34	3317	100	119.24	3317	100	3.17	3317	100	0.65
keller6	8062	2	3418.36	**7382**	–	–	8062	5	606.15	8062	44	1980.16
MANN_a9	372	100	0.01	372	100	$< \epsilon$	372	100	$< \epsilon$	NA	NA	NA
MANN_a27	**12277**	4	22864.81	**12264**	–	–	12281	1	88.28	12281	16	396.58
MANN_a45	**34194**	2	17524.05	**34129**	–	–	**34192**	1	390.58	34229	1	929.41
MANN_a81	**111137**	1	6167.28	**110564**	–	–	**111128**	1	832.24	111237	1	2942.54
hamming6-2	1072	100	$< \epsilon$	1072	100	$< \epsilon$	1072	100	$< \epsilon$	1072	100	$< \epsilon$
hamming6-4	134	100	$< \epsilon$	134	100	$< \epsilon$	134	100	$< \epsilon$	134	100	$< \epsilon$
hamming8-2	10976	100	0.80	10976	100	$< \epsilon$	10976	100	$< \epsilon$	10976	100	0.12
hamming8-4	1472	100	$< \epsilon$	1472	100	$< \epsilon$	1472	100	$< \epsilon$	1472	100	$< \epsilon$
hamming10-2	50512	67	24.47	50512	100	$< \epsilon$	50512	100	0.92	50512	100	6.64
hamming10-4	5129	8	32.49	**5086**	1	1433.07	5129	100	2.21	5129	100	26.86
gen200_p0.9_44	5043	100	0.02	5043	100	4.44	5043	100	$< \epsilon$	5043	100	0.01
gen200_p0.9_55	5416	100	0.43	5416	100	0.05	5416	100	0.33	5416	100	1.75
gen400_p0.9_55	6718	100	0.28	6718	2	340.11	6718	100	0.15	6718	2	0.18
gen400_p0.9_65	6940	100	0.11	**6935**	4	200.79	6940	100	0.04	6940	100	0.05
gen400_p0.9_75	8006	100	0.67	8006	100	$< \epsilon$	8006	100	0.88	8006	100	0.43
c-fat200-1	1284	100	0.01	1284	100	$< \epsilon$	1284	100	0.14	NA	NA	NA
c-fat200-2	2411	100	0.34	2411	100	$< \epsilon$	2411	100	0.06	NA	NA	NA
c-fat200-5	5887	100	0.20	5887	100	$< \epsilon$	5887	100	0.02	NA	NA	NA
c-fat500-1	1354	100	0.20	1354	100	$< \epsilon$	1354	100	0.73	NA	NA	NA
c-fat500-2	2628	100	3.10	2628	100	0.01	2628	100	0.33	NA	NA	NA
c-fat500-5	5841	100	1.15	5841	100	$< \epsilon$	5841	100	0.14	NA	NA	NA
c-fat500-10	11586	100	1.29	11586	100	$< \epsilon$	11586	100	0.06	NA	NA	NA
johnson8-2-4	66	100	$< \epsilon$	66	100	$< \epsilon$	66	100	$< \epsilon$	66	100	$< \epsilon$
johnson8-4-4	511	100	$< \epsilon$	511	100	$< \epsilon$	511	100	$< \epsilon$	511	100	$< \epsilon$

(Continued. . .)

Instance	BQP-PTS			PLS_W [31]			MS/TS [39]			BLS [5]		
	Best	Succ.	Time	Best	Succ.	Time	Best	Succ.	Time	Best	Succ.	Time
johnson16-2-4	548	100	$< \epsilon$	548	100	$< \epsilon$	548	100	0.23	548	100	0.01
johnson32-2-4	2033	40	26.71	2033	100	44.68	2033	100	0.53	2033	100	0.48
p_hat300-1	1057	100	0.03	1057	100	0.01	1057	100	0.02	1057	100	0.01
p_hat300-2	2487	100	0.02	2487	100	19.36	2487	100	$< \epsilon$	2487	100	0.02
p_hat300-3	3774	100	0.04	3774	47	418.11	3774	47	0.02	3774	47	0.01
p_hat500-1	1231	100	0.17	1231	100	0.42	1231	100	0.03	1231	100	0.04
p_hat500-2	**3920**	100	$< \epsilon$	3925	–	–	**3920**	100	$< \epsilon$	**3920**	100	0.01
p_hat500-3	5375	100	0.36	**5361**	–	–	5375	100	0.10	5375	100	0.05
p_hat700-1	1441	100	0.30	1441	100	0.20	1441	100	0.03	1441	100	0.01
p_hat700-2	5290	100	0.03	5290	100	78.51	5290	100	0.02	5290	100	0.02
p_hat700-3	7565	100	2.07	7565	12	718.40	7565	100	0.38	7565	100	0.13
p_hat1000-1	1514	100	3.78	1514	100	7.61	1514	100	0.08	1514	100	0.07
p_hat1000-2	5777	100	0.09	5777	87	940.62	5777	87	0.11	5777	87	0.04
p_hat1000-3	8111	100	0.65	**7986**	–	–	8111	100	1.23	8111	100	0.41
p_hat1500-1	1619	95	17.25	1619	100	48.91	1619	100	0.06	1619	100	0.14
p_hat1500-2	7360	100	3.61	**7328**	4	1056.19	7360	100	0.82	7360	100	0.18
p_hat1500-3	10321	9	34.14	**10014**	–	–	10321	96	188.38	10321	100	1.78
san200_0.7_1	3370	100	0.06	3370	100	$< \epsilon$	3370	100	0.17	3370	100	30.65
san200_0.7_2	2422	100	0.41	2422	66	397.38	2422	100	0.02	2422	100	0.01
san200_0.9_1	6825	100	0.02	6825	100	$< \epsilon$	6825	100	0.13	6825	100	23.68
san200_0.9_2	6082	100	0.02	6082	100	$< \epsilon$	6082	100	0.21	6082	100	0.19
san200_0.9_3	4748	100	0.64	4748	72	219.68	4748	72	$< \epsilon$	4748	100	0.02
san400_0.5_1	1455	100	5.74	1455	100	200.44	1455	100	0.06	1455	100	0.22
san400_0.7_1	3941	100	2.64	3941	100	0.03	3941	100	13.68	**3641**	98	–
san400_0.7_2	3110	100	6.81	3110	100	0.05	3110	100	43.34	3110	33	166
san400_0.7_3	2771	99	42.54	2771	100	4.41	2771	100	0.05	2771	100	0.05
san400_0.9_1	9776	100	0.31	9776	100	$< \epsilon$	9776	100	1.29	9776	100	6.25
san1000	1716	100	40.93	1716	–	–	1716	100	13.01	1716	100	4.94
sanr200-0.7	2325	100	0.08	2325	100	0.62	2325	100	$< \epsilon$	2325	100	0.01
sanr200-0.9	5126	100	$< \epsilon$	5126	5	182.54	5126	100	$< \epsilon$	5126	100	$< \epsilon$
sanr400-0.5	1835	100	1.41	1835	100	0.67	1835	100	0.02	1835	100	0.04
sanr400-0.7	2992	100	0.47	2992	100	141.50	2992	100	$< \epsilon$	2992	100	0.03

Table 1 presents the experimental results for the DIMACS-W benchmarks, where the columns under headings of BQP-PTS, PLS_W, MN/TS and BLS list respectively the best solution values $Best$ obtained by each algorithm, number of times to reach $Best$ over 100 runs $Succ.$, and the average CPU time $Time$ (in seconds) to reach $Best$. Notice that an entry with $< \epsilon$ signifies the average CPU time was less than 0.01 second and NA signifies the results are unavailable. The solution values inferior to the best known ones are marked in bold.

From Table 1, we observe that BQP-PTS obtains 76 best solutions for the evaluated 80 instances, better than 67 of PLS_W and competitive with 77 of MN/TS and 78 of BLS. For the 2 failed cases, BQP-PTS achieves the second best solutions. In addition, BQP-PTS has a success rate of 100% to reach the best solutions for 64 instances, 12 more than PLS_W but 4 and 5 less than MN/TS and BLS, respectively. Finally, BQP-PTS reaches the best known results within a reasonable time (less than 30 minutes) for most instances, except for 7 instances of C and MANN families. The long computing time for these instances could be attributed to their difficulty (in fact, the reference MVWCP heuristics also need longer time to attain their best solutions for these instances than for other instances). In particular, PLS_W can only attain its indicated best values for some of these C and MANN instances (as well as some other instances) under a long and relaxed time condition (indicated by '-' in Table 1). Moreover, unlike the dedicated MVWCP algorithms which incorporate problem specific implementation to ensure their search efficiency, BQP-PTS, as a general solver, does not benefit from such advantages.

Table 2 shows the results of the BQP-PTS approach compared to those of the MN/TS and BLS algorithms for the BHOSLIB-W benchmarks (the PLS_W algorithm does not report results for the BHOSLIB-W benchmarks).

Table 2 Computational comparisons of the BQP-PTS approach with the MS/TS and BLS algorithms on the set of BHOSLIB-W instances

Instances	BQP-PTS				MN/TS				BLS			
	Best	*Succ.*	*Avg*	Time	*Best*	*Succ.*	*Avg*	Time	*Best*	*Succ.*	*Avg*	Time
frb30-15-1	2990	100	2990	4.90	2990	100	2990	0.35	2990	100	2990	1.12
frb30-15-2	3006	100	3006	1.58	3006	100	3006	3.45	3006	100	3006	8.15
frb30-15-3	2995	100	2995	5.80	2995	100	2995	4.72	2995	100	2995	11.67
frb30-15-4	3032	100	3032	1.04	3032	100	3032	0.12	3032	100	3032	0.33
frb30-15-5	3011	100	3011	2.13	3011	100	3011	3.01	3011	100	3011	3.64
frb35-17-1	3650	100	3650	6.59	3650	100	3650	25.80	3650	100	3650	68.45
frb35-17-2	3738	100	3738	183.17	3738	96	3736.84	72.09	3738	100	3738	197.42
frb35-17-3	3716	100	3716	15.54	3716	100	3716	7.72	3716	100	3716	11.58
frb35-17-4	3683	100	3683	5.60	3683	77	3678.31	94.03	3683	100	3683	232.36
frb35-17-5	3686	100	3686	3.73	3686	100	3686	8.09	3686	100	3686	20.00
frb40-19-1	4063	100	4063	87.72	4063	83	4062.15	85.57	4063	96	4062.8	291.14
frb40-19-2	4112	100	4112	76.39	4112	87	4111.16	134.58	4112	100	4112	439.81
frb40-19-3	4115	100	4115	171.07	4115	19	4108.3	215.98	4115	46	4111.72	778.75
frb40-19-4	4136	100	4136	758.82	4136	89	4135.56	96.65	4136	98	4135.92	333.89
frb40-19-5	4118	100	4118	96.63	4118	90	4117.6	178.89	4118	88	4117.52	343.82
frb45-21-1	4760	100	4760	896.25	4760	44	4748.66	126.26	4760	58	4754.3	982.32
frb45-21-2	4784	100	4784	92.94	4784	47	4775.86	228.03	4784	100	4784	307.06
frb45-21-3	4765	100	4765	150.64	4765	26	4756.9	125.35	4765	88	4764.76	641.03
frb45-21-4	4799	100	4799	453.15	4799	43	4772.41	174.73	4799	96	4797.24	576.80
frb45-21-5	4779	100	4779	34.17	4779	82	4777.38	193.82	4779	100	4779	206.60
frb50-23-1	5494	20	5487.90	1911.49	5494	6	5484.74	186.62	5494	11	5486.41	1221.72
frb50-23-2	5462	15	5452.65	2338.40	5462	3	5434.14	149.66	5462	5	5440.22	2837.74
frb50-23-3	5486	100	5486	418.35	5486	53	5480.29	158.71	5486	98	5485.98	537.96
frb50-23-4	5454	28	5453.3	1957.22	5454	9	5451.69	176.41	5454	14	5453.14	1190.43
frb50-23-5	5498	100	5498	751.84	5498	89	5495.7	110.85	5498	100	5498	388.18
frb53-24-1	5670	43	5660.35	981.33	5670	5	5637.94	233.22	5670	13	5652.18	1056.82
frb53-24-2	5707	25	5694.3	1265.70	5707	6	5676.56	145.22	5707	3	5685.32	147.65
frb53-24-3	5640	90	5639.35	1486.24	5640	15	5610.79	215.79	5640	48	5629.38	984.53
frb53-24-4	5714	25	5700.75	1753.36	5714	7	5645.61	449.39	5714	13	5676.16	1604.50
frb53-24-5	5659	6	5653.05	2802.83	5659	5	5628.77	294.00	5659	4	5642.5	278.91
frb56-25-1	5916	19	5877.3	1035.00	5916	3	5836.85	308.90	5916	5	5860.82	1764.87
frb56-25-2	5886	3	5861.3	1428.18	**5872**	1	5807.7	73.25	5886	1	5838.96	1013.85
frb56-25-3	5859	1	5831.6	449.24	5859	1	5799.38	181.93	5859	1	5811	101.48
frb56-25-4	5892	5	5869.3	1756.22	5892	3	5839.16	104.58	5892	12	5860.86	1256.90
frb56-25-5	5853	1	5811.5	3549.57	**5839**	1	5768.39	322.70	5853	1	5787.04	4386.60
frb59-26-1	6591	67	6585.05	2228.21	6591	3	6547.53	166.20	6591	17	6571.6	1435.99
frb59-26-2	6645	40	6614.45	1820.56	6645	3	6567.07	212.49	6645	13	6602.34	1834.93
frb59-26-3	6608	1	6567.55	2561.16	6608	1	6514.18	232.77	6608	1	6542.74	507.93
frb59-26-4	6592	5	6533.5	3322.64	6592	1	6498.37	318.39	6592	6	6526.5	952.34
frb59-26-5	6584	9	6554.55	747.80	6584	1	6522.57	161.47	6584	5	6546.94	1512.09

Table 2 lists the best solution values *Best*, number of times hitting *Best* over 100 runs *Succ.*, the average solution values and the average CPU time *Time* (in seconds) to reach *Best* for each algorithm. From Table 2, we observe that BQP-PTS is able to attain the best known results for all the 40 instances as BLS does while MN/TS misses two best values (frb56-25-2 and frb56-25-5). In addition, BQP-PTS has a success rate of 100% to reach the best known results for 22 instances, better than MN/TS for 8 instances and BLS for 14 instances.

Moreover, BQP-PTS obtains better average solution values than MN/TS and BLS for 32 and 26 instances, while requiring slightly more computing time, particularly compared to MN/TS.

Finally, we also evaluated our BQP-PTS approach for the (unweighted) maximum clique instances. Without bothering to show tabulated results, we observed that BQP-PTS was able to attain the best known results for 77 of 80 DIMACS instances except for C2000.9 (79 vs 80), MANNa_45 (344 vs 345), MANNa_81 (1098 vs 1100) and for all the 40 BHOSLIB instances. Such a performance can be considered as quite good even compared to the best performing MCP algorithms presented in the recent review [41]. However, our BQP-PTS algorithm requires more computing time than specific MCP algorithms, in particular when it is applied to solve some very difficult instances.

5 Conclusion

We recast the maximum vertex weight clique problem (MVWCP) into the binary quadratic programming (BQP) model, which was solved by a Probabilistic Tabu Search algorithm. Experiments on two sets of challenging DIMACS-W and BHOSLIB-W benchmarks indicate that this general BQP approach is viable for solving the MVWCP problem. In particular, without incorporation of domain specific knowledge, this approach was able to attain the best known results for 76 out of 80 DIMACS-W instances and for all the 40 BHOSLIB-W instances within reasonable computing times. For the conventional maximum clique problem, the BQP approach achieved similar performances by attaining the best known results for 77 out of 80 DIMACS instances and for all the 40 BHOSLIB instances. However, our BQP approach is more time consuming than specific algorithms especially for some very difficult instances and some parameters need to be tuned to achieving its best performance. These computational outcomes demonstrate that the general BQP model constitutes an interesting alternative to solve these clique problems without calling for specific heuristics.

For future consideration, it would be interesting to explore using the Probabilistic Tabu Search design not only within the restart part of our method, but also periodically within the improving part of our method which currently consists of a relatively simple form of tabu search. Another interesting research line is to investigate automatic parameter tuning techniques to obtain a general and parameter free BQP solver.

Acknowledgment

We are grateful to the reviewers and the editors for their comments which help us to improve the paper. This work was supported by National Natural Science Foundation of China (Grant No. 71501157, 71371154) and Northwestern Polytechnical University (Grant No. 3102015RW007).

References

1. Alidaee B, Glover F, Kochenberger GA, Wang H (2007) Solving the maximum edge weight clique problem via unconstrained quadratic programming. European Journal of Operational Research 181:592-597
2. Alidaee B, Kochenberger GA, Lewis K, Lewis M, Wang H (2008) A new approach for modeling and solving set packing problem. European Journal of Operational Research 86(2):504-512
3. Babel L (1994) A fast algorithm for the maximum weight clique problem. Computing 52(1):31-38
4. Ballard D, Brown C (1983) Computer Vision.Prentice-Hall, Englewood Cliffs.
5. Benlic U, Hao JK (2013) Breakout local search for maximum clique problems. Computers & Operations Research 40(1):192–206
6. Bomze IM, Pelillo M, Stix V (2000) Approximating the maximum weight clique using replicator dynamics. IEEE Transactions on Neural Networks 11:1228-1241
7. Busygin S (2006) A new trust region technique for the maximum weight clique problem. Discrete Applied Mathematics 154:2080-2096
8. Carraghan R, Pardalos PM (1990) An exact algorithm for the maximum clique problem. Operations Research Letters 9(6):375–382
9. Dorigo M (1997) Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transaction on Evolutionary Computation 1(1):53-66
10. Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-Completeness. Freeman San Francisco, CA, USA
11. Glover F (1989) Tabu Search - Part I. ORSA Journal on Computing 1(3):190–206
12. Glover F, Hao JK (2010) Efficient evaluation for solving 0-1 unconstrained quadratic optimization problems. International Journal of Metaheuristics 1(1): 3-10
13. Glover F, Hao JK (2010) Fast 2-flip Move Evaluations for Binary Unconstrained Quadratic Optimization Problems. International Journal of Metaheuristics 1(2):100–107
14. Glover F, Laguna M (1997) Tabu Search. Kluwer Academic Publishers
15. Hansen P, Mladenović N (2001) Variable neighborhood search: Principles and applications. European Journal of Operational Research 130(3):449-467
16. He K, Huang W (2010) A quasi-human algorithm for solving the three-dimensional rectangular packing problem. SCIENCE CHINA Information Sciences 53(12): 2389-2398
17. Horst R, Pardalos PM, Thoai N.V. (1995) Introduction to Global Optimization, Nonconvex optimization and its applications, vol. 3, Kluwer Academic Publishers
18. Kochenberger GA, Glover F, Alidaee B, Rego C (2004) A unified modeling and solution framework for combinatorial optimization problems. OR Spectrum 26:237–250
19. Kochenberger G, Alidaee B, Glover F, Wang HB (2007) An effective modeling and solution approach for the generalized independent set problem. Optimization Letters 1:111-117
20. Kochenberger G, Hao JK, Lü Z, Wang H, Glover F (2013) Solving large scale max cut problems via tabu search. Journal of Heuristics 19(4):565-571
21. Kochenberger G, Hao JK, Glover F, Lewis M, Lü Z, Wang H, Wang Y (2014) The Unconstrained binary quadratic programming problem: a survey. Accepted to Journal of Combinatorial Optimization 28(1):58-81
22. Konc, J., Janězič, D. (2007). An improved branch and bound algorithm for the maximum clique problem. MATCH - Communications in Mathematical and in Computer Chemistry, 58, 569-590
23. Li C, Quan Z (2010) An efficient branch-and-bound algorithm based on MAXSAT for the maximum clique problem. In proceedings of the 24th AAAI conference on artificial intelligence, 128-133
24. Lewis M, Kochenberger G, Alidaee B (2008) A new modeling and solution approach for the set-partitioning problem. Computers & Operations Research 2008:807-813
25. Macreesh C, Prosser P (2013) Multi-threading a state-of-the-art maximum clique algorithm. Algorithms, 6(4):618-635
26. Manninno C, Stefanutti E (1999) An augmentation algorithm for the maximum weighted stable set problem. Computational Optimization and Applications 14:367-381
27. Östergård PRJ (2001) A new algorithm for the maximum weight clique problem. Nordic Journal of Computing 8(4):424-436

28. Östergård PRJ(2002) A fast algorithm for the maximum clique problem. Discrete Applied Mathematics 120(1): 197–207
29. Pajouh FM, Balasumdaram B, Prokopyev O (2013) On characterization of maximal independent sets via quadratic optimization. Journal of Heuristics 19(4):629-644
30. Pardalos PM, Rodgers GP (1992) A branch and bound algorithm for the maximum clique problem. Computers & Operations Research 19(5):363-375
31. Pullan W (2008) Approximating the maximum vertex/edge weighted clique using local search. Journal of Heuristics 14:117-134
32. Rebennack S, Oswald M, Theis D, Seitz H, Reinelt G, Pardalos PM (2011) A branch and cut solver for the maximum stable set problem. Journal of Combinatorial Optimization 21(4):434-457
33. Rebennack S, Reinelt G, Pardalos PM (2012) A tutorial on branch and cut algorithms for the maximum stable set problem. International Transactions in Operational Research 19(1-2):161-199
34. Segundo, P.S., Rodríguez-Losada, D., Jiménez, A. (2011). An exact bitparallel algorithm for the maximum clique problem. Computers & Operations Research, 38(2), 571–581.
35. Sengor NS, Cakir Y, Guzelis C, Pekergin F, Morgul O (1999) An analysis of maximum clique formulations and saturated linear dynamical network. ARI 51:268-276
36. Tomita E, Kameda T (2007) An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. Journal of Global Optimization 37(1):95–111
37. Wang Y, Lü Z, Glover F, Hao JK (2013) Probabilistic GRASP-Tabu Search Algorithms for the UBQP problem. Computers & Operations Research 40(12):3100-3107
38. Warren JS, Hicks IV (2006) Combinatorial branch-and-bound for the maximum weight independent set problem. Technical Report, Texas A&M University.
39. Wu Q, Hao JK, Glover F (2012) Multi-neighborhood tabu search for the maximum weight clique problem. Annals of Operations Research 196(1): 611-634
40. Wu Y, Huang W, Lau S, Wong CK, Young GH (2002) An effective quasi-human based heuristic for solving the rectangle packing problem. European Journal of Operational Research 141(2): 341-358
41. Wu Q, Hao JK (2015) A review on algorithms for maximum clique problems. European Journal of Operational Research 242:693-709
42. Xu JF, Chiu SY, Glover F (1996) Probabilistic tabu search for telecommunications network design. Combinational Optimization: Theory and Practice 1(1):69-94

Appendix

To illustrate the transformation from the MVWCP to the BQP, we consider the following graph:

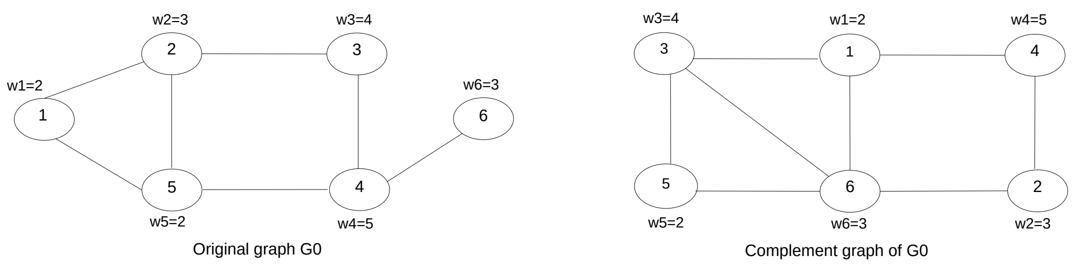

Fig. 1 A graph sample

Its linear formulation according to Eq. (1) is:

$$\begin{array}{rll} Max \quad f(x) = 2x_1 + 3x_2 + 4x_3 + 5x_4 + 2x_5 + 3x_6 & & \\ \text{s.t.} \quad x_1 + x_3 \leq 1; & & x_1 + x_4 \leq 1; \\ x_1 + x_6 \leq 1; & & x_2 + x_4 \leq 1; \\ x_2 + x_6 \leq 1; & & x_3 + x_5 \leq 1; \\ x_3 + x_6 \leq 1; & & x_5 + x_6 \leq 1. \end{array} \tag{6}$$

Choosing the scalar penalty $P = -15$, we obtain the following BQP model:

$$\begin{aligned} Max \quad f(x) = 2x_1 + 3x_2 + 4x_3 + 5x_4 + 2x_5 + 3x_6 - 30x_1x_3 - 30x_1x_4 \\ -30x_1x_6 - 30x_2x_4 - 30x_2x_6 - 30x_3x_5 - 30x_3x_6 - 30x_5x_6 \end{aligned} \tag{7}$$

which can be re-written as:

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{pmatrix} \times \begin{pmatrix} 2 & 0 & -15 & -15 & 0 & -15 \\ 0 & 3 & 0 & -15 & 0 & -15 \\ -15 & 0 & 4 & 0 & -15 & -15 \\ -15 & -15 & 0 & 5 & 0 & 0 \\ 0 & 0 & -15 & 0 & 2 & -15 \\ -15 & -15 & -15 & 0 & -15 & 3 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \tag{8}$$

Solving this BQP problem yields $x_3 = x_4 = 1$ (all other variables equal zero) and the optimal objective function value is 9.