Solving diversified top-k weight clique search problem with MaxSAT

Junping Zhou[1], **Chumin Li**[2], **Yupeng Zhou**[1], **Mingyang Li**[1], **Lili Liang**[1] and **Jianan Wang**[1⋆]

[1]School of Information Science and Technology, Northeast Normal University, Changchun, China
[2]Université de Picardie Jules Verne, France
zhoujp877@nenu.edu.cn, chu-min.li@u-picardie.fr,
{zhouyp605,limy344,liangll597,wangjn}@nenu.edu.cn

Abstract

Solving NP-complete problems with SAT solvers is an efficient approach for both academic and industrial problems. In this paper, we present two encoding strategies for solving the diversified top-k weight clique search (DTKWCS) problem and two specific practical applications of DTKWCS. Through several experiments we show that our encoding strategies are competitive, allowing to promote the applications of the DTKWCS problem, such as community detection, spectrum sharing, advertising placement, etc.

1 Introduction

Diversified top-k weight clique search (DTKWCS) is a problem that computes k cliques to maximize the sum of weights of all vertices contained in the cliques. This problem is NP-hard. It can be applied in spectrum sharing problem, advertising placement problem, gene expression and motif discovery, influential community search, sensor place problem, and anomaly detection in complex networks [Zheng *et al.*, 2011; Conrad *et al.*, 2010; Krause and Guestrin, 2007; Bao *et al.*, 2004; Bao *et al.*, 2016].

In solving DTKWCS in an unweighted graph, a trivial direct approach based on cliques enumeration is used [Feige, 1998]; however, the approach is time-consuming and not suitable for solving large graphs. Another direct-solving approach is proposed that can give approximate solutions [Long *et al.*, 2016]; however, the approach is not competitive in solving dense graphs and cannot guarantee the optimality of its solutions. Therefore, it is worth exploring a generic approach to solving DTKWCS.

In this paper, we provide a generic approach for solving DTKWCS, which is done by encoding the DTKWCS into the weighted partial MaxSAT (WPMS) problem and then solving WPMS with state-of-the-art solvers. It has been proven that solving NP problems, including academic and industrial problems, by encoding as a SAT instance or a WPMS instance is an efficient strategy [Chu *et al.*, 2010; Schwind *et al.*, 2016]. In this paper to perform the encoding of DTKWCS to WPMS, we present two encodings strategies: direct encoding (DE) and independent set partition based encoding (ISPE). The experimental results show that the two encoding strategies are competitive.

2 Preliminaries

G=(V, E, w) is an undirected weighted graph, where V and E are sets of vertices and edges respectively, and w is a weight function that assigns a non-negative integer, called *weight* (or *cost*), to each vertex v. A clique c_i of a graph G is a subset of vertices in G such that every two distinct vertices in the subset are adjacent.

A literal is either a Boolean variable (variable for brevity in the rest of paper) x or its negation $\neg x$. A clause is a disjunction of literals, which is satisfied if and only if at least one literal in it taking the value *true*. A weighted clause is a pair (c, w), where c is a clause, and w is the weight of the clause. A weighted clause is *hard* if its weight is infinite; otherwise, the clause is *soft*. A WPMS formula F in CNF is a conjunction of hard and soft clauses. The purpose of the WPMS problem is to find a truth assignment for F by satisfying all hard clauses and then maximizing the sum of weights of all satisfied soft clauses.

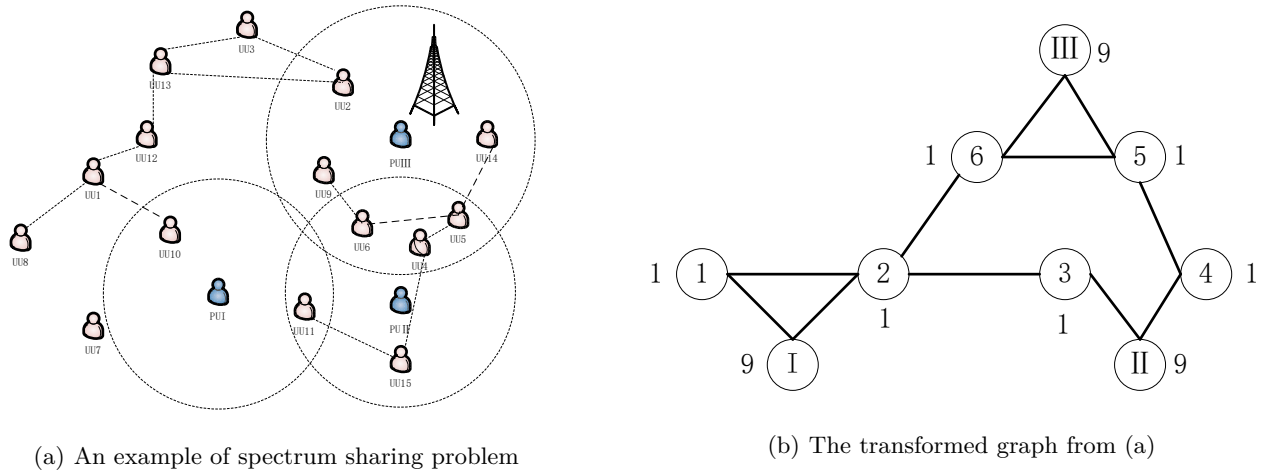
The diversified top-k weight clique search problem is to compute k cliques to maximize the sum of weights of all vertices contained in the cliques; that is, $\Sigma_{v\in\{c_1\cup c_2\cup\cdots\cup c_k\}}w(v)$ is maximized by given a weighted graph G and an integer k, where c_i is one of the k cliques, and $w(v)$ is the weight of the vertex v in G.

3 Applications

This section presents two concrete practical applications of the DTKWCS problem.

3.1 Spectrum sharing problem

We start with a wireless network application. With the rapid development of wireless network technologies, limited spectrum resources become the main bottleneck restricting the performance of wireless networks. Recently, spectrum utilization can be increased by exploiting cognitive radio networks, which allows unlicensed



(a) An example of spectrum sharing problem

(b) The transformed graph from (a)

Figure 1: The spectrum sharing problem and its transformed graph.

users to employ the spectrum resources occupied by primary users. Suppose that a cognitive radio network contains R primary users, N unlicensed users, and k spectrum channels. The primary users employ the fixed channel allocation regulation, so that any two primary users occupy different channels. Each primary user i has a coverage of radium d_i. If the distance from a primary user i to an unlicensed user j is less than d_i, the unlicensed user j cannot use the channel occupied by the nearby primary user i because the two users would interfere with each other. Similarly, each unlicensed user also has a coverage of radium. Two unlicensed users cannot use the same channel when their interference areas overlap.

The spectrum sharing problem is to allocate available channels to maximize network utilization. When the maximum number of available channels is limited, the problem is to perform spectrum assignment to unlicensed users so that the maximum number of unlicensed users can use channels simultaneously. This problem can be solved by reducing it into the DTKWCS problem, mapping users, including primary users and unlicensed users, into vertices, adding an edge between two users that can share a channel, and assigning weight 1 to each unlicensed user, and weight $R+N$ to each primary user. Then the spectrum allocation problem is equivalent to find k cliques to maximize system utilization.

Example 1. Figure 1(a) describes an example of spectrum sharing problem, where a cognitive radio network contains 3 primary users (PUI-PUIII), 6 unlicensed users (UU1-UU6), and 3 spectrum channels marked A, B, and C. The primary user I, II, and III use channels A, B, and C respectively. Their coverages are shown as dotted circles around each primary user and the conflicted unlicensed users are connected with dotted lines. Then, the cognitive radio network can be abstracted into a graph illustrated in Figure 1(b). The numbers labelled outside each vertex are the weights of the corresponding vertices. This spectrum allocation instance is equivalent to the diversified top-3 weight clique search problem. We can find 3 cliques {1, 2, I}, {5, 6, III}, {3, II}, which maximizes the total weights of covered vertices. Thus, the channel A is assigned to unlicensed users 1, 2; the channel B is to 3, and the channel C is to 5, 6.

3.2 Advertising placement problem

Advertisement on TV is a common way for companies to publicize their products. How to select programmes from TV to advertise to catch maximum viewers is extremely important for these companies. Suppose that a company plans to advertise in k parallel sessions simultaneously, where a session is a set of programmes that are not mutually exclusive with respect of time scheduling. The advertising placement problem is to find k sessions to maximize the sum of viewers of the programmes in k sessions. This problem can be also modeled as DTKWCS problem, mapping programmes into vertices, adding an edge between two programmes without time duration overlap, and assigning the number of viewers as weight to each programme. From the modeling process, we can see the vertices in a clique constitute a session. Then the advertising placement problem is equivalent to select k cliques to maximize the sum of viewers of the programmes in these k cliques.

Example 2. Figure 2(a) describes an example of advertising placement problem, where the programmes are from programme1 to programme5, and the numbers of viewers of the corresponding programmes are recorded in the brackets. Then, the problem can be transformed into a graph illustrated in Figure 2(b). The numbers labelled outside of each vertex are the weights of the corresponding vertices. When $k = 2$, the instance is equivalent to the diversified top-2 weight clique search problem. We can find 2 cliques {1, 3, 5} and {1, 2, 4, 5}, which can maximize the sum of weights of covered vertices by the 2 cliques. Thus, the company can select these 2 sessions

{1, 3, 5} and {1, 2, 4, 5}, i.e., programmes {1, 2, 3, 4, 5} to publicize their products.

4 Encodings from DTKWCS to WPMS

In this section, we present two encodings from DTKWCS to WPMS. The first one is a direct encoding, and the second one is based on independent set partition.

4.1 Direct encoding

In this subsection, we propose a novel encoding, which we call the direct encoding (DE), from DTKWCS to WPMS. The basic idea of the DE is derived from the following observations. First, because the DTKWCS problem requires to find k cliques, we encode each vertex into k variables; that is, the vertex v_i is expanded into the variables $x_{i1}, x_{i2}, ..., x_{ik}$. Thus, the variable $x_{ij} = true$ if and only if the vertex v_i is in the j^{th} clique. Second, the DTKWCS and WPMS problems are both used to compute a solution to maximize the sum of weights of vertices (or soft clauses). Then, DE encoding creates hard clauses that could guarantee every feasible solution of a WPMS instance to form k cliques. Finally, the DE encoding employs a direct way to encode soft clauses; that is, each vertex v_i defines a soft clause, which is satisfied if and only if v_i is in at least one of the k cliques.

Formally, given a graph $G = (V, E, w)$ and an integer k, we define the DE encoding as follows.

1. For each $v_i \in V$, create k variables $x_{i1}, x_{i2}, \ldots, x_{ik}$.
2. For any two unconnected vertices v_i and v_j in V (i.e., $(v_i, v_j) \notin E$), create k hard clauses: $(\neg x_{i1} \vee \neg x_{j1}, \infty)$, $(\neg x_{i2} \vee \neg x_{j2}, \infty)$, ..., $(\neg x_{ik} \vee \neg x_{jk}, \infty)$.
3. For each vertex $v_i \in V$, create a soft clause $(x_{i1} \vee x_{i2} \vee \cdots \vee x_{ik}, w(v_i))$.

We denote the resulting WPMS formula by ϕ. The DE encoding has the following properties.

- Any feasible solution of ϕ, that is, any truth assignment satisfying all hard clauses of ϕ, gives k cliques. To see this, let us partition the variables assigned with the value *true* into the k subsets: $\{x_{i_1 1}, x_{i_1 2}, x_{i_1 3}, \ldots\}$, $\{x_{i_2 1}, x_{i_2 2}, x_{i_2 3}, \ldots\}$, ..., $\{x_{i_k 1}, x_{i_k 2}, x_{i_k 3}, \ldots\}$. Any two vertices corresponding to two variables in a subset, saying $v_{i_{j1}}$ and $v_{i_{j2}}$, must be adjacent; otherwise, a hard clause $\neg x_{i_{j1} j} \vee \neg x_{i_{j2} j}$ was created because $v_{i_{j1}}$ and $v_{i_{j2}}$ were not adjacent, which would be falsified.
- Any k cliques give a truth assignment satisfying all hard clauses of ϕ: $x_{ij} = true$ if and only if the vertex v_i is in the j^{th} clique.
- Any optimal solution of ϕ gives k cliques covering the vertices with the maximum sum of weights. To see this, note that each soft clause corresponds to a unique vertex, and it is satisfied if and only if the corresponding vertex is covered. Thus, the maximum sum of weights of the satisfied soft clauses is equal to the maximum sum of weights of the covered vertices.
- DE ensures the cliques found by WPMS solvers are as disjoint as possible. That is because the aim of the WPMS problem is to detect a truth assignment to maxmize the sum of weights of all satisfied soft clauses and each satisfied soft clause reprsents a covered vertex.

Example 3. Figure 3 displays a graph G=(V, E, w), where $w(v_1) = 3, w(v_2) = 4, w(v_3) = 2, w(v_4) = 1$. Let $k = 2$. The WPMS instance encoding DTKWCS using DE consists of the hard clauses $(\neg x_{11} \vee \neg x_{31}, \infty)$, $(\neg x_{12} \vee \neg x_{32}, \infty)$, $(\neg x_{21} \vee \neg x_{41}, \infty)$, and $(\neg x_{22} \vee \neg x_{42}, \infty)$, because there are two pairs of non-adjacent vertices $< v_1, v_3 >$ and $< v_2, v_4 >$, and soft clauses $(x_{11} \vee x_{12}, 3)$, $(x_{21} \vee x_{22}, 4)$, $(x_{31} \vee x_{32}, 2)$, and $(x_{41} \vee x_{42}, 1)$. An optimal assignment of the WPMS instance is $\{x_{11} = false, x_{21} = true, x_{31} = true, x_{41} = false, x_{12} = true, x_{22} = false, x_{32} = false, x_{42} = true\}$, which satisfies all soft clauses. From this solution, we can conclude that the solution of the DTKWCS instance is $\{\{v_2, v_3\}, \{v_1, v_4\}\}$ covering all vertices of G.

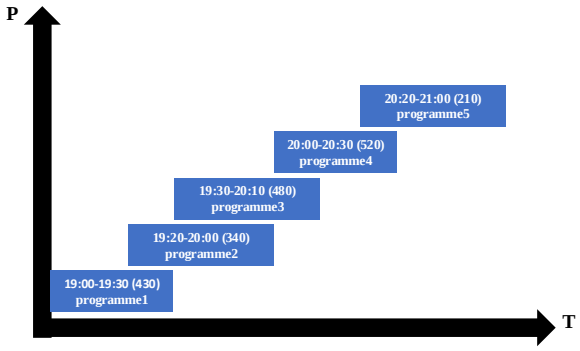
4.2 Independent set partition based encoding

After presenting a straight encoding, we introduce another novel encoding, called Independent Set Partition based Encoding (ISPE). The ISPE includes two conversion processes: reducing a DTKWCS instance into a new version of partial MaxSAT, named literal WPMS (LWPMS), and subsequent transforming LWPMS into WPMS. Before the introduction of the ISPE, some related definitions are given. LWPMS is a conjunction of hard clauses and literal-weighted soft clauses. The literal-weighted soft clause is composed of weighted literals denoted by (l, w), where l is a literal, and w is the weight of the literal. Given a graph G, we note that an independent set is a set of disconnected vertices.

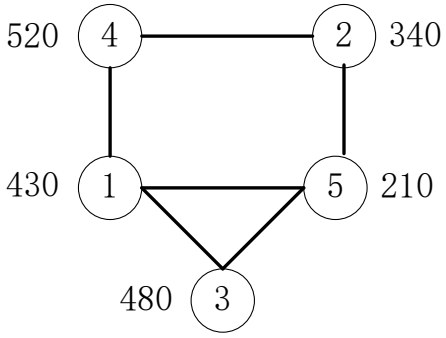
Next, we present the first part of ISPE from DTKWCS into LWPMS by given a graph $G = (V, E, w)$ and an integer k as follows.

1. For each $v_i \in V$, create k variables $x_{i1}, x_{i2}, \ldots, x_{ik}$.
2. For any two unconnected vertices v_i and v_j in V (i.e., $(v_i, v_j) \notin E$), create k hard clauses: $(\neg x_{i1} \vee \neg x_{j1}, \infty)$, $(\neg x_{i2} \vee \neg x_{j2}, \infty)$, ..., $(\neg x_{ik} \vee \neg x_{jk}, \infty)$.
3. For each $v_i \in V$, create $\binom{k}{2}$ hard clauses. Specifically, for any two variables x_{ir} and $x_{ij} (r \neq j, 1 \leqslant r, j \leqslant k)$ generated by v_i, create a hard clause $(\neg x_{ir} \vee \neg x_{ij}, \infty)$.
4. Partition the graph G into several disjoint independent sets, and ensure that the vertices in the disjoint independent sets constitute V. Then for each independent set $\{v_i, v_j, \ldots, v_r\}$, create k literal-weighted soft clauses $(x_{is}, w(v_i)) \vee (x_{js}, w(v_j)) \vee ... \vee (x_{rs}, w(v_r))$ $(s = 1, 2, \ldots, k)$.

The following is the intuition behind the first part of ISPE. The hard clauses generated by the disconnected vertices guarantee that the vertices build up k cliques.



(a) An example of advertising placement problem



(b) The transformed graph from (a)

Figure 2: The advertising placement problem and its transformed graph.

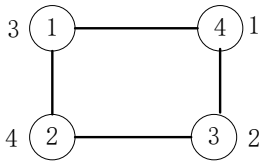


Figure 3: A graph G with 4 vertices and 4 edges.

To ensure that the k cliques are not duplicated, we generate $\binom{k}{2}$ hard clauses for each vertex. Furthermore, literal-weighted soft clauses guarantee that the sum of weights of covered vertices is maximum. We partition the graph into independent sets in a predetermined ordering. Suppose that the current independent sets are $S_1, S_2, ..., S_r$ (in this order, r is 0 at the beginning of the partition process), the current first vertex v is inserted into the first S_i such that v is non-adjacent to all vertices already in S_i. If such a S_i does not exist, a new independent set S_{i+1} is opened and v is inserted.

In view that no existing solvers can solve the LWPMS, we manage to encode LWPMS into WPMS in the second part. By comparing LWPMS and WPMS, we understand that the difference between both is the type of soft clauses: literal weighted and clause weighted. Therefore, we need to transform the literal-weighted soft clauses into clause-weighted soft clauses (i.e., soft clauses). The method of the conversion is done by iteratively splitting the weighted literals. For each literal-weighted soft clause, we generate a set of soft clauses, as shown in Algorithm 1. After encoding all literal-weighted soft clauses into soft clauses, the LWPMS is reduced into WPMS. Similarly, $x_{ij} = 1$ if and only if the vertex v_i is in the j^{th} clique. The optimal solution of the WPMS instance corresponds to the maximum total weights of the covered vertices.

Example 4. Let us also consider the graph G=(V, E, w) demonstrated in Figure 3. Given $k = 2$, the ISPE first reduces the DTKWCS instance into LWPMS,

Algorithm 1: To-Soft-Clause

Input: a literal-weighted soft clause $(x_{is}, w(v_i)) \vee (x_{js}, w(v_j)) \vee ... \vee (x_{rs}, w(v_r))$
Output: a set of soft clauses C

```
W ← {w(v_i), w(v_j), ..., w(v_r)};
L ← {x_is, x_js, ..., x_rs};
δ ← min W;
while L ≠ ∅ do
    add a soft clause (c, δ) to C, where c is a
      disjunction of all literals in L ;
    delete the weights equivalent to δ from W and
      the literals whose weight is equal to δ from L;
    W is updated by each weight in W minus δ;
    δ ← min W;
return C;
```

which consists of the hard clauses $(\neg x_{11} \vee \neg x_{12}, \infty)$, $(\neg x_{21} \vee \neg x_{22}, \infty)$, $(\neg x_{31} \vee \neg x_{32}, \infty)$, $(\neg x_{41} \vee \neg x_{42}, \infty)$, $(\neg x_{11} \vee \neg x_{31}, \infty)$, $(\neg x_{12} \vee \neg x_{32}, \infty)$, $(\neg x_{21} \vee \neg x_{41}, \infty)$, $(\neg x_{22} \vee \neg x_{42}, \infty)$, and the literal-weighted soft clauses $(x_{11}, 3) \vee (x_{31}, 2)$, $(x_{21}, 4) \vee (x_{41}, 1)$, $(x_{12}, 3) \vee (x_{32}, 2)$, and $(x_{22}, 4) \vee (x_{42}, 1)$. Then the ISPE encodes the LWPMS into WPMS, mainly encoding the literal-weighted soft clauses into soft clauses based on Algorithm 1. The reduced soft clauses are $(x_{11} \vee x_{31}, 2)$, $(x_{11}, 1)$, $(x_{21} \vee x_{41}, 1)$, $(x_{21}, 3)$, $(x_{12} \vee x_{32}, 2)$, $(x_{12}, 1)$, $(x_{22} \vee x_{42}, 1)$, and $(x_{22}, 3)$. The hard clauses in the WPMS instance have no change. Then the optimal assignment of the WPMS instance is $\{x_{11} = false, x_{21} = true, x_{31} = true, x_{41} = false, x_{12} = true, x_{22} = false, x_{32} = false, x_{42} = true\}$. From this solution, we obtain that the solution of the DTKWCS instance is $\{\{v_2, v_3\}, \{v_1, v_4\}\}$ covering all vertices of G.

5 Evaluation

To test the DE and ISPE, we perform experiments using the approximate DTKWCS solver EnumKOpt [Long *et al.*, 2016], the exact WPMS solver RC2-2018 [Ignatiev *et al.*, 2019], and the heuristic WPMS solver TT-Open-

Table 1: The comparison results on sparse graphs.

Instance	k	avg(s)	TT-Open-wbo-inc			RC-2018	
			EnumKOpt	DE	ISPE	DE	ISPE
			time(avg(s))	time(avg(s))	time(avg(s))	time	time
V40	1	3.07	**0(3.07)**	**0(3.07)**	**0(3.07)**	0.0002	0.0001
	2	6.67	**0(6.67)**	**0(6.67)**	**0(6.67)**	0.0664	0.0002
	3	9.00	**0(9.00)**	**0(9.00)**	**0(9.00)**	1.7557	0.0005
	4	11.60	**0(11.60)**	**0(11.60)**	**0(11.60)**	52.7998	0.0011
	5	13.97	0(13.96)	**0(13.97)**	**0(13.97)**	-	0.0028
	6	16.10	**0(16.10)**	**0(16.10)**	**0(16.10)**	-	0.0108
V50	1	3.07	**0(3.07)**	**0(3.07)**	**0(3.07)**	0.0004	0.0001
	2	6.07	**0(6.07)**	**0(6.07)**	**0(6.07)**	0.2199	0.0004
	3	9.03	**0(9.03)**	**0(9.03)**	**0(9.03)**	10.0517	0.0007
	4	11.93	**0(11.93)**	**0(11.93)**	**0(11.93)**	230.2293	0.0012
	5	14.70	**0(14.70)**	**0(14.70)**	**0(14.70)**	-	0.0024
	6	17.27	0(17.20)	0(17.20)	**0(17.27)**	-	0.0062
V60	1	3.17	**0(3.17)**	**0(3.17)**	**0(3.17)**	0.0006	0.0003
	2	6.17	**0(6.17)**	**0(6.17)**	**0(6.17)**	0.5761	0.0008
	3	9.17	**0(9.17)**	**0(9.17)**	**0(9.17)**	58.8047	0.0013
	4	12.17	**0(12.17)**	**0(12.17)**	**0(12.17)**	-	0.0022
	5	15.17	**0(15.17)**	**0(15.17)**	**0(15.17)**	-	0.0029
	6	18.17	**0(18.17)**	**0(18.17)**	0(18.10)	-	0.0051
V70	1	3.37	**0(3.37)**	**0(3.37)**	**0(3.37)**	0.0007	0.0003
	2	6.53	**0(6.53)**	**0(6.53)**	**0(6.53)**	1.1819	0.001
	3	9.57	**0(9.57)**	**0(9.57)**	**0(9.57)**	167.7378	0.0019
	4	12.57	**0(12.57)**	**0(12.57)**	**0(12.57)**	-	0.003
	5	15.57	**0(15.57)**	**0(15.57)**	**0(15.57)**	-	0.0045
	6	18.57	**0(18.57)**	**0(18.57)**	**0(18.57)**	-	0.0065
V80	1	3.40	**0(3.40)**	**0(3.40)**	**0(3.40)**	0.0008	0.0005
	2	6.47	**0(6.47)**	**0(6.47)**	**0(6.47)**	2.2178	0.0017
	3	9.47	**0(9.47)**	**0(9.47)**	**0(9.47)**	-	0.0031
	4	12.47	**0(12.47)**	**0(12.47)**	**0(12.47)**	-	0.0048
	5	15.47	**0(15.47)**	**0(15.47)**	**0(15.47)**	-	0.0171
	6	18.47	**0(18.47)**	**0(18.47)**	**0(18.47)**	-	0.0098
V90	1	3.70	**0(3.70)**	**0(3.70)**	**0(3.70)**	0.0011	0.0007
	2	6.83	**0(6.83)**	**0(6.83)**	**0(6.83)**	3.6216	0.0025
	3	9.87	**0(9.87)**	**0(9.87)**	**0(9.87)**	-	0.0048
	4	12.90	**0(12.90)**	0(12.87)	**0(12.90)**	-	0.01
	5	15.90	**0(15.90)**	0(15.87)	**0(15.90)**	-	0.0106
	6	18.90	**0(18.90)**	0(18.87)	**0(18.90)**	-	0.0152
V100	1	3.93	**0(3.93)**	**0(3.93)**	**0(3.93)**	0.0016	0.001
	2	7.37	**0(7.37)**	**0(7.37)**	**0(7.37)**	5.4904	0.0033
	3	10.47	**0(10.47)**	**0(10.47)**	**0(10.47)**	-	0.0068
	4	13.50	0(13.47)	0(13.47)	**0(13.50)**	-	0.0103
	5	16.50	**0(16.50)**	0(16.47)	**0(16.50)**	-	0.0138
	6	19.50	**0(19.50)**	0(19.43)	**0(19.50)**	-	0.0216

Table 2: The comparison results on graphs with 60 vertices.

Instance	k	avg(s)	TT-Open-wbo-inc			RC-2018	
			EnumKOpt	DE	ISPE	DE	ISPE
			time(avg(s))	time(avg(s))	time(avg(s))	time	time
P10	1	3	**0(3.00)**	**0(3.00)**	**0(3.00)**	0.0005	0.0003
	2	6	**0(6.00)**	**0(6.00)**	**0(6.00)**	0.5917	0.0008
	3	9	**0(9.00)**	**0(9.00)**	**0(9.00)**	83.4188	0.0016
	4	12	**0(12.00)**	**0(12.00)**	**0(12.00)**	-	0.0023
	5	15	**0(15.00)**	**0(15.00)**	**0(15.00)**	-	0.0033
	6	18	**0(18.00)**	**0(18.00)**	**0(18.00)**	-	0.0042
P20	1	4	**0(4.00)**	**0(4.00)**	**0(4.00)**	0.0006	0.0008
	2	8	**0(8.00)**	**0(8.00)**	**0(8.00)**	1.9699	0.0021
	3	12	**0(12.00)**	**0(12.00)**	**0(12.00)**	-	0.0031
	4	16	**0(16.00)**	**0(16.00)**	**0(16.00)**	-	0.01
	5	20	**0(20.00)**	**0(20.00)**	**0(20.00)**	-	0.0142
	6	23	**0(23.00)**	**0(23.00)**	**0(23.00)**	-	0.1251
P30	1	5	**0(5.00)**	**0(5.00)**	**0(5.00)**	0.0012	0.0015
	2	10	**0(10.00)**	**0(10.00)**	**0(10.00)**	4.0429	0.003
	3	15	**0(15.00)**	**0(15.00)**	**0(15.00)**	-	0.0152
	4	20	**0(20.00)**	**0(20.00)**	**0(20.00)**	-	0.0112
	5	25	-	0(24.00)	**0(25.00)**	-	0.013
	6	29	-	0(28.00)	**0(29.00)**	-	0.1004
P40	1	6	**0(6.00)**	**0(6.00)**	**0(6.00)**	0.0022	0.0033
	2	12	0(11.30)	**0(12.00)**	**0(12.00)**	7.0075	0.0078
	3	18	0(17.10)	**0(18.00)**	**0(18.00)**	-	0.0545
	4	24	-	0(23.00)	**0(24.00)**	-	0.0289
	5	29	-	0(28.00)	**0(29.00)**	-	0.1885
	6	34	-	0(33.00)	**0(34.00)**	-	179.0345
P50	1	8	**0(8.00)**	**0(8.00)**	**0(8.00)**	0.0021	0.0025
	2	16	0(15.07)	**0(16.00)**	**0(16.00)**	5.9974	0.007
	3	22	-	**0(22.00)**	**0(22.00)**	-	0.0979
	4	28	-	**0(28.00)**	**0(28.00)**	-	0.504
	5	34	-	**0(34.00)**	0(33.00)	-	164.7289
	6	40	-	**0(39.00)**	**0(39.00)**	-	34.284
P60	1	15	**0(15.00)**	**0(15.00)**	**0(15.00)**	0.0079	0.0063
	2	30	-	**0(30.00)**	**0(30.00)**	3.9801	1.4185
	3	42	-	**0(42.00)**	**0(42.00)**	230.6567	68.5837
	4	52*	-	**0(52.00)**	**0(52.00)**	-	-
	5	59	-	**0(59.00)**	**0(59.00)**	6.4364	2.1628
	6	60	-	**0(60.00)**	**0(60.00)**	0.0008	0.0885
P70	1	23	0(22.20)	**0(23.00)**	**0(23.00)**	0.0022	0.0013
	2	41	-	**0(41.00)**	**0(41.00)**	0.881	1.3745
	3	55	-	**0(55.00)**	**0(55.00)**	0.5056	0.6349
	4	60	-	**0(60.00)**	**0(60.00)**	0.001	0.0117
	5	60	-	**0(60.00)**	**0(60.00)**	0.0013	0.0213
	6	60	-	**0(60.00)**	**0(60.00)**	0.0011	0.0932

wbo-Inc [Fahiem *et al.*, 2019]. These solvers are considered to be the best in their category. RC2-2018 and TT-Open-wbo-Inc are executed on WPMS instances obtained using the encoding presented in this paper, while EnumKOpt is directly executed on graphs. The cut off time of RC2-2018 and TT-Open-wbo-Inc is 1800 and 60 seconds, respectively. Since EnumKOpt is an approximation algorithm, it has no cut off time and is executed until it ends itself. In addition, it returns a fault when it fails to solve an instance. Furthermore, for each instance we run TT-Open-wbo-Inc 10 times and record the average value of the 10 runs as the final solution. In the experimental results, the time is measured in seconds. When the time is marked "0", it represents that the instance can be solved in less than 0.0001 seconds. If a solver fails to find a feasible solution in the cut off time, we mark "-". In all experiments, k varies from 1 to 6. We implement the two encodings in the C++ programming language. All experiments are performed on a workstation under Linux ubuntu 18.04 with Intel(R) Core i7-7700 3.60 GHz CPU and 32 GB RAM.

5.1 Comparion on sparse graphs

Table 1 shows the performance of the three solvers on sparse graphs in which two vertices are adjacent with probability 0.1. The number of vertices of these graphs ranges from 40 to 100, the weight of all vertices is assigned 1 because EnumKOpt can only solve DTKWCS on unweighted graphs. For each scale of graphs, we generate 50 instances. In the table, Vi represents the graphs containing i vertices, $avg(s)$ records the mean sum of weights of covered vertices in a set of 50 instances found by the exact solver RC2-2018, and $time(avg(s))$ records the average time and average size of these instances found by the corresponding solvers. As can be seen from the table, EnumKOpt can efficiently obtain the optimal solution except V40 ($k = 5$), V50 ($k = 6$), and V100 ($k = 4$). The instances encoded by DE can be derived nearly optimal solution by TT-Open-wbo-Inc and fast solved by RC2-2018 when k is small. The instances encoded by ISPE can be obtained good-quality solutions by TT-Open-wbo-Inc, the quality of which is higher than EnumKOpt. Furthermore, RC2-2018 solves these instances always in a very short time. Thus, the experiment indicates that when solving DTKWCS on sparse graphs, ISPE is effective and efficient.

5.2 Comparion on graphs with different densities

We compare the three solvers by varying the probability with which two vertices are adjacent in a graph in Table 2. In the experiment, the number of vertices in all graphs is 60, the weight of all vertices is 1, and the probability is varied from 0.1 to 0.7. We also generate 50 instances for graphs with each probability. We see that the larger the probability is, the denser the graphs. As in Table 2, Pi represents that the probability that any

Table 3: The comparison results on graphs from BHOSLIB.

Instance	k	size	EnumKOpt	TT-Open-wbo-inc		RC-2018	
				DE	ISPE	DE	ISPE
			time(size)	time(size)	time(size)	time	time
frb30-15-1	1	30	-	**0(30)**	0.01(28)	0.3157	-
(450, 83198)	2	54.00*	-	**0.01(54)**	0.01(53)	-	-
	3	78.00*	-	0.01(75)	**0.01(78)**	-	-
	4	99.00*	-	0.02(92)	**0.02(99)**	-	-
	5	121.00*	-	0.01(113)	**0.02(121)**	-	-
	6	143.00*	-	0.01(130)	**0.01(143)**	-	-
frb30-15-2	1	30	-	**0(30)**	**0(30)**	3.7318	3.0665
(450, 83151)	2	59	-	0.01(53)	**0.01(55)**	-	95.211
	3	78.00*	-	0.01(74)	**0(78)**	-	-
	4	101.00*	-	0.01(91)	**0.01(101)**	-	-
	5	123.00*	-	0.01(111)	**0.01(123)**	-	-
	6	150.00*	-	0.02(131)	**0.01(150)**	-	-
frb30-15-3	1	30	-	**0(30)**	**0(30)**	0.1344	3.4444
(450, 83216)	2	59	-	0.01(54)	**0(55)**	-	16.8359
	3	79.00*	-	0.01(74)	**0.01(79)**	-	-
	4	101.00*	-	0.02(91)	**0.01(101)**	-	-
	5	125.00*	-	0.01(112)	**0.01(125)**	-	-
	6	148.00*	-	0.01(131)	**0.01(148)**	-	-
frb30-15-4	1	30	-	**0(30)**	**0(30)**	0.0392	276.7267
(450, 83194)	2	59	-	0.01(54)	**0(55)**	-	117.4003
	3	79.00*	-	0.01(73)	**0.01(79)**	-	-
	4	101.00*	-	0.01(93)	**0.01(101)**	-	-
	5	125.00*	-	0.01(111)	**0.01(125)**	-	-
	6	145.00*	-	0.01(131)	**0.01(145)**	-	-
frb30-15-5	1	30	-	**0(30)**	**0(30)**	0.0793	6.9701
(450, 83231)	2	59	-	0.01(54)	**0(56)**	-	22.4641
	3	79.00*	-	0.02(75)	**0.01(79)**	-	-
	4	102.00*	-	0.02(92)	**0.01(102)**	-	-
	5	127.00*	-	0.01(111)	**0.01(127)**	-	-
	6	149.00*	-	0.01(134)	**0.01(149)**	-	-
frb40-19-1	1	40	-	**0.01(40)**	**0.01(40)**	0.638	8.0238
(760, 247106)	2	72.00*	-	**0.01(72)**	**0.01(72)**	-	-
	3	111.00*	-	0.01(96)	**0.03(111)**	-	-
	4	140.00*	-	0.02(119)	**0.04(140)**	-	-
	5	170.00*	-	0.03(143)	**0.03(170)**	-	-
	6	202.00*	-	0.04(172)	**0.04(202)**	-	-
frb40-19-2	1	40.00*	-	**0.01(40)**	0(39)	126.3444	-
(760, 247157)	2	73.00*	-	0.01(72)	**0(73)**	-	-
	3	105.00*	-	0.02(95)	**0.01(105)**	-	-
	4	137.00*	-	0.03(118)	**0.02(137)**	-	-
	5	166.00*	-	0.04(145)	**0.03(166)**	-	-
	6	197.00*	-	0.03(172)	**0.03(197)**	-	-
frb40-19-3	1	40	-	**0.01(38)**	0(37)	6.5005	-
(760, 247325)	2	70.00*	-	0.02(70)	**0.01(70)**	-	-
	3	101.00*	-	0.02(95)	**0.01(101)**	-	-
	4	131.00*	-	0.04(116)	**0.02(131)**	-	-
	5	162.00*	-	0.03(143)	**0.03(162)**	-	-
	6	192.00*	-	0.03(168)	**0.02(192)**	-	-
frb40-19-4	1	39	-	**0.01(39)**	0.01(37)	-	-
(760, 246825)	2	71.00*	-	**0.01(71)**	0.01(68)	-	-
	3	101.00*	-	0.01(96)	**0.02(101)**	-	-
	4	127.00*	-	0.03(118)	**0.02(127)**	-	-
	5	155.00*	-	0.03(144)	**0.03(155)**	-	-
	6	184.00*	-	0.03(169)	**0.02(184)**	-	-
frb40-19-5	1	40	-	**0(39)**	0(38)	140.2172	-
(760, 246801)	2	71.00*	-	0.01(70)	**0.01(71)**	-	-
	3	101.00*	-	0.02(98)	**0.01(101)**	-	-
	4	133.00*	-	0.04(119)	**0.01(133)**	-	-
	5	164.00*	-	0.03(144)	**0.02(164)**	-	-
	6	194.00*	-	0.03(170)	**0.02(194)**	-	-

Table 4: The results on advertising placement instances.

Instance	k	size	TT-Open-wbo-inc		RC-2018	
			DE	ISPE	DE	ISPE
			time(size)	time(size)	time	time
P10	1	1628	**0(1628)**	**0(1628)**	0.0001	0.0001
	2	1628	**0(1628)**	**0(1628)**	0.0001	0.0002
	3	1628	**0(1628)**	**0(1628)**	**0.00**	0.0002
	4	1628	**0(1628)**	**0(1628)**	0.0001	0.0003
	5	1628	**0(1628)**	**0(1628)**	0.0001	0.0004
	6	1628	**0(1628)**	**0(1628)**	0.0001	0.0007
P20	1	2341	**0(2341)**	**0(2341)**	0.0002	0.0002
	2	2617	**0(2617)**	**0(2617)**	0.0008	0.0004
	3	2845	**0(2845)**	**0(2845)**	0.0007	0.0006
	4	2970	**0(2970)**	**0(2970)**	0.0026	0.0009
	5	3051	**0(3051)**	**0(3051)**	0.0054	0.0015
	6	3122	**0(3122)**	**0(3122)**	0.0051	0.0017
P30	1	2572	**0(2572)**	**0(2572)**	0.0005	0.0003
	2	3109	**0(3109)**	**0(3109)**	0.0035	0.001
	3	3556	**0(3556)**	**0(3556)**	0.0102	0.0017
	4	3783	**0(3783)**	**0(3783)**	0.1363	0.01
	5	3983	**0(3983)**	**0(3983)**	2.2837	16.0011
	6	4132	**0(4132)**	**0(4132)**	15.2011	-
P40	1	3674	**0(3674)**	**0(3674)**	0.0004	0.0005
	2	4508	**0(4508)**	**0(4508)**	0.0071	0.0017
	3	5045	**0(5045)**	**0(5045)**	0.0186	0.0032
	4	5272	**0(5272)**	**0(5272)**	0.1675	0.14
	5	5472	**0(5472)**	**0(5472)**	1.0632	-
	6	5621	**0(5621)**	**0(5621)**	19.1466	-
P50	1	4133	**0(4133)**	**0(4133)**	0.001	0.0006
	2	5828	**0(5828)**	**0(5828)**	0.0053	0.0015
	3	6422	**0(6422)**	**0(6422)**	0.0211	0.039
	4	6814	**0(6814)**	**0(6814)**	0.1362	26.0454
	5	7128	**0(7128)**	**0(7128)**	0.6038	-
	6	7328	**0(7328)**	**0(7328)**	16.2458	-
P60	1	4575	**0(4575)**	**0(4575)**	0.0021	0.0008
	2	6562	**0(6562)**	**0(6562)**	0.0141	0.0057
	3	7656	**0(7656)**	**0(7656)**	0.0246	0.422
	4	8319	**0(8319)**	**0(8319)**	0.3307	-
	5	8766	**0(8766)**	**0(8766)**	2.6617	-
	6	9111	**0(9111)**	**0(9111)**	37.5346	-
P70	1	5525	**0(5525)**	**0(5525)**	0.0026	0.0006
	2	8085	**0(8085)**	**0(8085)**	0.0139	0.0056
	3	9223	**0(9223)**	**0(9223)**	0.0439	0.0804
	4	9886	**0(9886)**	**0(9886)**	0.4195	-
	5	10333	**0(10333)**	**0(10333)**	4.0618	-
	6	10678	**0(10678)**	**0(10678)**	25.2137	-
P87	1	5926	**0(5926)**	**0(5926)**	0.0039	0.001
	2	9519	**0(9519)**	**0(9519)**	0.0268	0.0102
	3	11040	**0(11040)**	**0(11040)**	0.0506	0.3139
	4	11990	**0(11990)**	**0(11990)**	1.244	-
	5	12570	**0(12570)**	**0(12570)**	13.8274	-
	6	12961	**0(12961)**	**0(12961)**	76.0653	-

two vertices have an edge is i. The $avg(s)$ also records the mean sum of weights of covered vertices in a set of 50 instances, where the number with $*$ is the best solution found by TT-Open-Wbo-Inc, the others are found by RC2-2018. $time(avg(s))$ records the average time and average size of these instances found by the corresponding solvers. Among the results in Table 2, we observe the performance of EnumKOpt becomes worse when the probability is increased, whereas the performance of the other two solvers are stable. This indicates that our encodings are effective on dense graphs and it is better for DTKWCS to be solved by converting to WPMS using DE and ISPE when handling dense graphs.

5.3 Comparion on BHOSLIB graphs

The results showed in Table 3 summarize the comparison of the three solvers on a subset of BHOSLIB graphs [max, 2020]. In the experiment, the weight of all vertices is assigned 1. In Table 3, the numbers below the name of each instance are $(|V|, |E|)$, where $|V|$ and $|E|$ represent the number of vertices and edges respectively. The *size* records the sum of weights of covered vertices of each graph, where the number with $*$ is the best solution found by TT-Open-Wbo-Inc, the others are found by RC2-2018. From the table, we see that EnumKOpt fails for these graphs; RC2-2018 can obtain the optimal solution for a few instances, whereas TT-Open-wbo-Inc can always give a feasible solution. Moreover, the graphs encoded by ISPE always be produced a better quality solution than the graphs encoded by DE, which both use the same solver TT-Open-Wbo-Inc. It indicates that ISPE is efficient for solving hard graphs for the graphs in BHOSLIB, which are famous for its hardness.

5.4 Comparion on advertising placement problem

We conduct an experiment on the advertising placement problem. Since EnumKOpt is designed for unweighted DTKWCS, we compare the two WPMS solvers TT-Open-wbo-Inc and RC2-2018 on a set of real size TV programmes provided by SAPPRFT. All instances are available on the website [1]. The number of vertices of these instances ranges from 10 to 87. In Table 4, the $size \times 10000$ is the sum of viewers of selected programmes. From the results shown in Table 4, we see that almost all instances encoded by DE and ISPE can be obtained the optimal solutions solved by TT-Open-wbo-Inc. Moreover, the instances encoded by DE have a better performance than ISPE solved by RC2-2018, which indicates that the DE encoding is effective for solving the advertisement putting problem.

6 Conclusion

This paper defines two encoding strategies for solving the DTKWCS problem into the WPMS problem. It can be noted that DE encoding is a direct way, whereas ISPE is based on independent set partition. The experimental results show that our encoding strategies are efficient and effective, which also remedy the lack of dedicated exact solvers for the DTKWCS problem.

References

[Bao *et al.*, 2004] Y. Bao, S. Wang, and et al. Yan, B. Emergent clique formation in terrorist recruitment. In *Proc the AAAI-04 Workshop on Agent Organizations: Theory and Practice*, 2004.

[Bao *et al.*, 2016] Y. Bao, S. Wang, and et al. Yan, B. Research on maximal weighted independent set-based graph coloring spectrum allocation algorithm in cognitive radio networks. In *Proc the International Conference on Communications, Signal Processing and Systems*, 2016.

[Chu *et al.*, 2010] Min Li Chu, Felip Manyà, Quan Zhe, and Zhu Zhu. Exact minsat solving. In *International Conference on Theory & Applications of Satisfiability Testing*, pages 363–368, 2010.

[Conrad *et al.*, 2010] L. Conrad, M. Aaron, R. Fergal, and Neil H. Detecting highly overlapping community structure by greedy clique expansion. In *Proc the 4th SNA–KDD Workshop on Social Network Mining and Analysis*, pages 112–119, 2010.

[Fahiem *et al.*, 2019] B. Fahiem, J. Matti, and M. Ruben. Maxsat evaluation 2019 : Solver and benchmark descriptions. 2019.

[Feige, 1998] Uriel Feige. A threshold of ln n for approximating set cover. *JOURNAL OF THE ACM*, 45:314–318, 1998.

[Ignatiev *et al.*, 2019] Alexey Ignatiev, Antonio Morgado, and Joao Marquessilva. Rc2: an efficient maxsat solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 11(1):53–64, 2019.

[Krause and Guestrin, 2007] Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *Proc AAAI Conference on Artificial Intelligence*, pages 22–26, 2007.

[Long *et al.*, 2016] Long, Yuan, Lu, Qin, Xuemin, Lin, Lijun, Chang, Wenjie, and Zhang. Diversified top-k clique search. *Vldb Journal*, 25:171–196, 2016.

[max, 2020] Maxsat evaluation 2020. https://networkrepository.com/bhoslib.php, 2020.

[Schwind *et al.*, 2016] Nicolas Schwind, Tenda Okimoto, Maxime Clement, and Katsumi Inoue. Representative solutions for multi-objective constraint optimization problems. In *International Conference on Principles of Knowledge Representation & Reasoning*, pages 601–604, 2016.

[Zheng *et al.*, 2011] Xiaoqi Zheng, Taigang Liu, Zhongnan Yang, and Jun Wang. Large cliques in arabidopsis gene coexpression network and motif discovery. *Journal of Plant Physiology*, 168(6):611–618, 2011.

[1] http://ai.nenu.edu.cn/yinmh/index.html