mathematics

MDPI

Article

A Restart Local Search for Solving Diversified Top-*k* Weight Clique Search Problem

Jun Wu and Minghao Yin *

Information Science and Technology, Northeast Normal University, Changchun 130117, China; wuj342@nenu.edu.cn

* Correspondence: ymh@nenu.edu.cn

Abstract: Diversified top-*k* weight clique (DTKWC) search problem is an important generalization of the diversified top-*k* clique (DTKC) search problem with practical applications. The diversified top-*k* weight clique search problem aims to search *k* maximal cliques that can cover the maximum weight in a vertex weighted graph. In this work, we propose a novel local search algorithm called TOPKWCLQ for the DTKWC search problem which mainly includes two strategies. First, a restart strategy is adopted, which repeated the construction and updating processes of the maximal weight clique set. Second, a scoring heuristic is designed by giving different priorities for maximal weight cliques in candidate set. Meanwhile, a constraint model of the DTKWC search problem is constructed such that the research concerns can be evaluated. Experimental results show that the proposed algorithm TOPKWCLQ outperforms than the comparison algorithm on large-scale real-world graphs.

Keywords: CPLEX; DTKWC; local search; MILP; scoring strategy

Citation: Wu, J.; Yin, M. A Restart Local Search for Solving Diversified Top-*k* Weight Clique Search Problem. *Mathematics* **2021**, *9*, 2674. https://doi.org/10.3390/math9212674

Academic Editors: Ismael González Yero and Dorota Kuziak

Received: 24 September 2021
Accepted: 19 October 2021
Published: 21 October 2021

1. Introduction

Given an undirected graph $G = (V, E)$, a clique is a subset of the graph G, where any two vertices are adjacent. The maximal clique (MC) is a clique with the largest cardinality in the graph G. The maximum weight clique (MWC) is a generalization of MC with a positive integer assigned to each vertex as its weight value. The diversified top-*k* clique (DTKC) search problem aims to find a set with at most *k* maximal cliques to occupy as many vertices as possible, where *k* is a parameter that requires to be provided. The diversified top-*k* weight clique (DTKWC) search problem [1] attempts to search a set with at most *k* maximal weight cliques in the graph G with the largest total weight of covered vertices in these cliques, which can be readily verified as a NP-hard problem [2].

The MC and related problems have lots of applications, especially in real-world applications such as combinatorial auction [3], community detection [4,5] and video object segmentation [6]. Recently, considerable attentions have also been paid to solve top-*k* problems on large graphs [2,7,8]. This kind of problem can be very well applied to practical applications, such as the influential community [9], motif discovery in molecular biology [10]. For example, citation networks are usually represented as a type of social network with papers and links between citation relationships. In citation networks, denoted as graph G, papers are considered as vertices, and citation relationships are the edges between papers. The influence on the paper is viewed as a weight in G. The problem aims to search the top-*k* maximal divisive groups with different domains in G, which can be regarded as finding a DTKWC solution.

To solve MC and WMC problems on large-scale graphs effectively, some related methods have been proposed. These algorithms are usually divided into two categories: (1) the first one is the exact algorithms which can guarantee the optimality of the solutions, such as [11–13]. But exact algorithms may fail to solve the graphs within a reasonable time when the scale of them are larger. The second one is the local search algorithm, which is considered to find a suboptimal solution within a reasonable time for medium

even larger graphs. And a large amount of effort has been devoted to designing different local search algorithms. For example, there exist lots of local search algorithms for solving WMC, e.g., [14,15]. Although there exist many algorithms to solve MC and WMC problems, currently, there are very few methods for diversified top-k cohesive groups. Such as Yuan et al. [7,8] (2015, 2016) proposed the concept of DTKC and then provided an approximate algorithm for it. Wu et al. [2] (2020) provides a local search algorithm to solve the DTKC search problem in large graphs, and it is state-of-the-art algorithm for DTKC search problem. Wu and Yin [16] (2021) introduce a problem of finding cohesive groups, named DTKSP problem, and develop a local search method based on some new heuristic strategies for this problem. Zhou et al. [1] (2021) encode the DTKWC search problem into the weighted partial MaxSAT (WPMS) problem, including direct encoding (DE) and independent set partition based encoding (ISPE), and solving WPMS with state-of-the-art solvers. However, this method is limited to solve real-world large graphs, because it is failed to encode large graphs into WPMS.

In this work, we propose a local search algorithm for the DTKWC search problem in large graphs, which provides a local optimal solution within a reasonable time and avoids the generation and storage of all maximal weight cliques. It aims at addressing the aforementioned problem. This algorithm, named TOPKWCLQ (which stands for top-k weight cliques), is based on two main strategies.

The first strategy is a restart method that can deal with the cycling problem. In the process of searching for the maximal weight clique, TOPKWCLQ will repeat to create a new maximal weight clique after initializing the set of maximal weight cliques and update this set through a scoring function. When the algorithm cannot be updated at fixed steps, it performs the restart process with the current best candidate solution.

The second strategy is a scoring function, which is designed by giving different priorities for maximal weight cliques in the candidate solution. During the searching process, TOPKWCLQ constructs and then maintains a candidate solution which size is at most k by adding or removing the maximal weight cliques according to the score value of each one. The score of each maximal weight clique is calculated by the total weight of the vertices that the clique has exclusively in the candidate solution.

To date, there is no suitable comparable algorithm for the DTKWC search problem on large scale of real-world graphs. Thus, we compare TOPKWCLQ with a commercial solver, CPLEX solver, with the constraint formulas proposed in this paper. Extensive performance experiments are executed to demonstrate that our proposed algorithm can achieve both high effectiveness and efficiency on real-world large-scale graphs.

The remainder of the paper is organized as follows. In Section 2, we propose the necessary background knowledge about diversified top-k weight clique search problem and formalize the DTKC and DTKWC search problem. In Section 3, we describe the TOPKWCLQ algorithm and the techniques it implements. In Section 4, we report extensive experimental results to demonstrate DTKWC's high performance compared to CPLEX with our model in solving the DTKWC search problem, and finally, the conclusions are given in Section 5.

2. Diversified Top-k Weight Clique Search Problem

In this section, some notations and basic definitions which are applied to the DTKWC search problem are introduced. Then the proof of NP-hardness about DTKWC search problem is given. Next, the constraint formulas which are used in CPLEX solver as the mathematical model for DTKC and DTKWC search problem are proposed, respectively.

2.1. Definition and Notations

A *weighted graph* $G = (V, E, w)$ is a graph including $|V|$ vertices and $|E|$ edges, w is a weight function that assigns to each vertex v_i of V a non-negative integer $w(v_i)$ representing its weight. $v_i^{w(v_i)}$ represents that vertex v_i has weight $w(v_i)$.

Definition 1 (**Maximal clique (MC)**). *Given an unweighted graph $G(V, E)$, a clique c in G is a set of vertices such that for any $u \in G$, $v \in c$ $(u \neq v)$, we have $(u, v) \in E$. A clique c in G called a maximal clique if there exists no clique $c^{'}$ in G such that $c \subset c^{'}$.*

Definition 2 (**Maximal weight clique (MWC)**). *Given a weighted graph $G(V, E, w)$, a weight clique c in G is a set of vertices such that for any $u \in G$, $v \in c$ $(u \neq v)$, we have $(u, v) \in E$ and the weight of c is $\omega(c) = \sum_{v_i \in c} w(v_i)$. A weight clique c in G called a maximal weight clique if there exists no clique $c^{'}$ in G such that $\omega(c) < \omega(c^{'})$.*

Given a set of maximal cliques $C = \{c_1, c_2, \dots\}$, the *coverage* of C, denoted by $cov(C)$, is the set of vertices covered by C, i.e., $cov(C) = \bigcup_{c_i \in C} c_i$.

Definition 3 (**Diversified top-k clique (DTKC)**). *Given an unweighted graph $G(V, E)$ and an integer k, the problem of diversified top-k clique search is to compute a set C, such that each $c \in C$ is a maximal clique, $|C| \leq k$, and $cov(C)$ is maximized. C is called diversified top-k cliques.*

Given a set of maximal (weighted) cliques $C = \{c_1, c_2, \dots\}$, the *private vertices* of a maximal (weighted) clique c in C, denoted by $priv(c, C)$, are a subset of vertices of c not contained in any other clique in C, i.e., $priv(c, C) = c \setminus cov(C \setminus c)$. The *weight* of C is the total weight of the set of vertices in G covered by the cliques in C, denoted by $W(C)$, as below

$$W(C) = \sum_{v_i \in (\bigcup_{c_j \in C} c_j)} w(v_i) \tag{1}$$

For the DTCK search problem, $w(v_i) = 1$, $i \in [1, |V|]$. The *overlapping* of C, denoted by $overlap(S)$, is a set of vertices that are covered by maximal cliques in C more than once.

Definition 4 (**Diversified top-k weight clique (DTKWC)**). *Given a weighted graph $G(V, E, w)$ and an integer k, the problem of diversified top-k weight clique search is to compute a set C, such that each $c \in C$ is a maximal weight clique, $|C| \leq k$, and $W(C)$ is maximized. C is called diversified top-k weight cliques.*

2.2. Constraint Formulation for DTKWC Search Problem

The DTKWC search problem is a generalization of the DTKC search problem [2] which aims to find a maximal clique set with at most k size with maximum total weight and a lower overlapping among all possible maximal clique sets from a given graph. Hence, we first give the formulas of the DTKC search problem and then expand them to the formulas of the DTKWC search problem. The DTKC search problem can be formulated as a mixed integer linear program (MILP) as follows:

$$\text{OBJ1}: \ Maximum \quad W_1(G) = \sum_{i \in [1,|V|]} X_i \tag{2}$$

$$\text{OBJ2}: \ Minimum \quad W_2(G) = \frac{\sum_{h=1}^{k} \sum_{i \in [1,|V|]} x_{ih} - \sum_{i \in [1,|V|]} X_i}{k-1} \tag{3}$$

Subject to:

$$x_{ih} + x_{jh} \leq 1, \quad \forall (i, j) \in \bar{E}, 1 \leq h \leq k \tag{4}$$

$$X_i \leq \sum_{i \in [1,|V|]} x_{ih}, \quad \forall i \in [1, |V|], 1 \leq h \leq k \tag{5}$$

$$x_{ih} \in \{0, 1\}, \quad \forall i \in [1, |V|], 1 \leq h \leq k \tag{6}$$

$$X_i \in \{0, 1\}, \quad \forall i \in [1, |V|] \tag{7}$$

where x_{ih} is the binary variable associated with the vertex i, such that $x_{ih} = 1$ if vertex v_i is in the h'th maximal clique, $x_{ih} = 0$ otherwise. X_i is also a binary variable associated

with vertex i. $X_i = 1$ if there exists a vertex i in a maximal clique, $X_i = 0$ otherwise. And constraint (4) is guaranteed that there is an edge between every two vertices in a clique. Constraint (5) means there is one clique including vertex v_i, then $X_i = 1$. Constraints (6) and (7) give the range of the variables.

According to the formulas above, we give the MILP of the DTKWC search problem below:

$$\text{OBJ1}: \ Maximum \quad W_1(G) = \sum_{i \in [1,|V|]} (X_i w_i) \tag{8}$$

$$\text{OBJ2}: \ Minimum \quad W_2(G) = \frac{\sum_{h=1}^{k} \sum_{i \in [1,|V|]} (x_{ih} w_i) - \sum_{i \in [1,|V|]} (X_i w_i)}{k-1} \tag{9}$$

Subject to:

$$x_{ih} + x_{jh} \leq 1, \quad \forall (i,j) \in \bar{E}, 1 \leq h \leq k \tag{10}$$

$$X_i \leq \sum_{i \in V} x_{ih}, \quad \forall i \in [1,|V|], 1 \leq h \leq k \tag{11}$$

$$x_{ih} \in \{0,1\}, \quad \forall i \in [1,|V|], 1 \leq h \leq k \tag{12}$$

$$X_i \in \{0,1\}, \quad \forall i \in [1,|V|] \tag{13}$$

Similarly, x_{ih} and X_i represent the binary variables corresponding with the vertex i. $x_{ih} = 1$, if vertex i appears in the h'th maximal clique, $x_{ih} = 0$ otherwise. $X_i = 1$, if vertex v_i belongs to any maximal clique of C, $X_i = 0$ otherwise. w_i denotes the weight of the vertex i. Constraints (10)–(13) have the same intentions as the above constraints (4)–(7), respectively.

In the above formulas, both of these two problems aim to minimize the value of objective "OBJ1" on basis of maximizing the value of objective "OBJ2". Thus, we can obtain the optimal solution of DTKC (DTKWC) search problem.

3. TOPKWCLQ: A Local Search Method for the DTKWC Search Problem

In this section, we will outline the framework of our algorithm. We use a restart strategy that interleaves between the construction and updating processes of the maximal weight clique set to enhance the quality of the candidate solution.

The restart procedure of the local search avoids the previous trajectory but turns to explore more different maximal weight clique sets. We construct these different maximal weight clique sets by combing the maximal weight cliques constructed from different starting vertices at each iteration. Thus, in the DTKWC search problem, using this restart strategy, TOPKWCLQ can improve the quality of the current candidate solution step by step.

At each restart iteration, we need to construct a new maximal weight clique one by one and eliminate the original maximal weight clique with the scoring function from the current candidate solution until no further improvement is found in the limited updating steps or the limit time is out. Thus, it can save the search time of a single iteration and restart the algorithm as soon as possible. After the updating procedure, a current candidate solution, that is, a local optimal solution, can be found and the algorithm will update the solution by comparing this local optimal solution with the maintained candidate solution from the previous iterations. Finally, until the time limit runs out, the algorithm returns to a maximal weight clique set as the best solution.

In the following, we will provide a random restart local search algorithm for the DTKWC search problem, called TOPKWCLQ.

3.1. Maximal Weight Clique Scoring Function

Before describing the algorithm framework, we first give a core issue in the algorithm TOPKWCLQ to evaluate the priorities of each maximal weight clique. During the search process, TOPKWCLQ must maintain a maximal weight clique set of size at most k as a candidate solution of the DTKWC search problem. Therefore, it is important to balance the quality and efficiency of the solutions by determining which maximal weight clique

should be included or eliminated from the current candidate solution. For this reason, we will define a scoring function based on the total weight of its private vertices presented in Section 2 for each maximal weight clique during the updating process.

Definition 5 (Score function (score(c))). *Given a weighted graph $G = (V, E, w)$, a maximal weight clique set C and a maximal weight clique c_i of G ($c_i \in C$). We use $score(c_i)$ to define the benefit of c_i after adding a maximal weight clique to the set C. The score of c in C is defined as*

$$score(c_i) = \sum_{v_j \in priv(c_i, C)} w(v_j) \tag{14}$$

The maximal weight clique selection method used in the *UpdateSolution* procedure is based on this scoring function. It attempts to determine the eliminated maximal weight clique with the smallest *score* value by computing the scoring function for each maximal weight clique in the candidate solution C after adding a new maximal weight clique into C.

3.2. TOPKWCLQ Algorithm: The Top-Level Algorithm

The proposed TOPKWCLQ algorithm (see the flowchart in Figure 1) combines an initialization procedure aiming to generate a feasible initial solution and a local search procedure aiming at improving the initial solution. The top level of TOPKWCLQ is outlined in Algorithm 1, as described below.

Algorithm 1 TOPKWCLQ($G, k, cutoff$)

```
1:  Input: an weighted graph G(V, E, W), one integer k, cutoff time
2:  Output: a set C* containing at most k maximal weight cliques
3:  m ← m_0, C* ← ∅; /* m_0 is a parameter used in BMS strategy */
4:  while (elapsed time < cutoff) do
5:     RemainingSet ← V;
6:     /* m_max is another parameter used in BMS strategy */
7:     if (m < m_max) then
8:        m ← 2 * m;
9:     else
10:       m_0 ← m_0 + 1; m ← m_0;
11:    end if
12:    C ← InitKCliques(G, m, RemainingSet);
13:    if (cov(C) = V) then
14:       return C;
15:    end if
16:    C ← LocalSearch(G, C, m, RemainingSet);
17:    if (W(C*) < W(C)) then
18:       C* ← C;
19:    end if
20: end while
21: return C* ;
```

First, we introduce the basic framework of our algorithm, which is presented in Algorithm 1. A current best global solution C^* will be initialized as an empty set. Then the TOPKWCLQ starts a loop until the limited time reaches the maximum which equals $cutoff$ (lines 4–20). Before this loop, the parameters m_0 and m_{max} of the best from multiple selection (BMS) strategy which is used in [2] to solve the DTKC search problem were given first (lines 6–11) and update the value of m in the loop. Then the TOPKWCLQ adopts a function $InitKCliques$ to construct enough maximal weight cliques as an initialization solution (line 12). After the initialization procedure, if $cov(C)$ equals to V, TOPKWCLQ will return C as a candidate solution (lines 13–15); Otherwise, update the current candidate solution C by using $LocalSearch$ method (line 16). If the total weight of the vertices in

C^* is smaller than the total weight of C, that is $W(C^*) < W(C)$, then replace C^* with C (lines 17–19). When the elapsed time is bigger than the cutoff time, TOPKWCLQ stops searching and returns C^*.

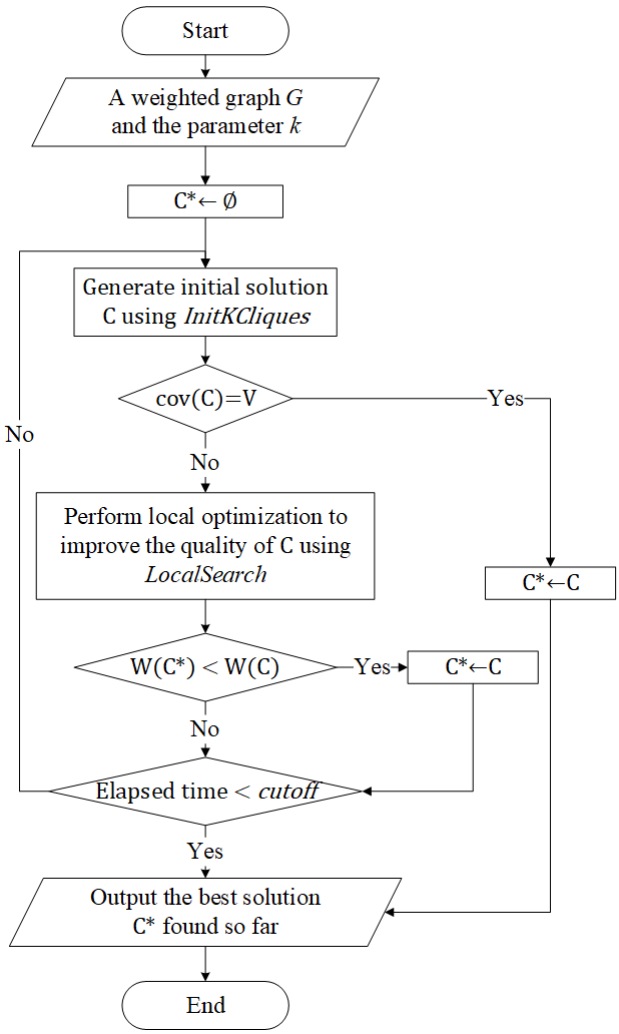

Figure 1. The main flowchart of the proposed TOPKWCLQ algorithm.

In this section, the technical details of the TOPKWCLQ algorithm are introduced. The function to create a maximal weight clique from a random vertex is introduced in Section 3.3 . In Section 3.4, the initialization procedure is presented. Section 3.5 presents the local search updating procedure of our algorithm.

3.3. Constructing a Maximal Weight Clique with Diversity

At each stage of our algorithm, we need constantly to find different maximal weight cliques to add into the candidate solution. Therefore, we design a method called *GetClique* which uses the vertices in *RemainingSet* to construct the maximal weight cliques according to the properties of the DTKWC search problem. Let *Candset* denote the vertices which are adjacent to all vertices already in c. We also design a function $b[v]$ which will be utilized during the initialization procedure to represent the benefit of a vertex v, the expression is as follows,

$$b[v] = \sum_{u \in (N(v) \cap Candset)} w(u). \tag{15}$$

The Algorithm 2 shows the pseudo-code of *GetCliques*. First, c is initialized as an empty set. Then, *GetCliques* iteratively and randomly selects a vertex from *RemainingSet*

which includes all vertices in V but excludes the vertices in the current candidate solution. If the set of *RemainningSet* is empty (line 2), then the algorithm returns c, and c is empty, which means we cannot create one more maximal weight clique. Otherwise, *GetClique* selects a vertex v from *RemainingSet* randomly and then adds it to the set c. Then, the algorithm adds all neighbours of v into *CandSet*. If *CandSet* is not empty, *GetClique* will find a maximal weight clique by the BMS strategy (which is proposed by [17]) used to select the better next vertex as the added vertex to the current partial clique (lines 7–22). In this situation, if the cardinality of *CandSet* is smaller than the parameter m, the algorithm will pick a vertex v from *CandSet* with the greatest $\hat{b}$, breaking ties in favour of the older one; Otherwise, *GetClique* selects the vertex with biggest benefit from m vertices that randomly selects from *Candset*. After that, we can get a better result by just calculating the score of at most m vertices. *CandSet* is updated for selecting the next vertex of the maximal weight clique.

Algorithm 2 GetClique$(G, m, RemainingSet)$

1: $c \leftarrow \emptyset$;
2: **if** $(RemainingSet = \emptyset)$ **then**
3: **return** c;
4: **end if**
5: $v \leftarrow$ randomly select a vertex from *RemainingSet*;
6: $c \leftarrow \{v\}$, $CandSet \leftarrow \{u | u \in N(v)\}$;
7: **while** $(CandSet \neq \emptyset)$ **do**
8: **if** $(|CandSet| < m)$ **then**
9: pick the vertex v from *CandSet* with the greatest $\hat{b}$, breaking ties in favour of the older one;
10: **else**
11: $v \leftarrow$ randomly select a vertex from *CandSet*;
12: **for** $(iter := 1$ to $m - 1)$ **do**
13: $v' \leftarrow$ randomly select a vertex from *CandSet*;
14: **if** $(\hat{b}[v'] > \hat{b}[v])$ **then**
15: $v \leftarrow v'$, $\hat{b}[v] \leftarrow \hat{b}[v']$;
16: **end if**
17: **end for**
18: **end if**
19: $c \leftarrow c \cup \{v\}$;
20: remove v from *RemainingSet*;
21: $CandSet \leftarrow CandSet \cap N(v)$;
22: **end while**
23: **return** c

3.4. The Initialization Procedure

In this subsection, we will explain the initialization procedure, which is outlined in Algorithm 3. It is the first stage of our algorithm. At the beginning of this procedure, a current candidate solution C is set to empty. Due to the DTKWC search problem needs to find a solution which is a set including at most k maximal weight cliques, this method attempts to create the maximal weight cliques randomly through *GetClique* that introduced in the above subsection. If we get an empty result from *GetClique* until there is no more vertex not belongs to C or we have created k maximal weight cliques. In this method, the starting vertices are generated randomly from a set that includes the vertices never used as a starting vertex. Repeat this random process, we can get the diversified maximal weight cliques that do not depend on the corresponding information acquired during the previous process. Moreover, starting from an unvisited vertex to construct the solution of the DTKWC search problem will overcome the cycling problem, i.e., revisiting the same solution within a short time in the local search algorithm.

Algorithm 3 InitKCliques($G, m, RemainingSet$)

```
1: C ← ∅;
2: while (|C| < k) do
3:    if (cov(C) = V) then
4:       return C;
5:    end if
6:    /* create a maximal weight clique with BMS strategies */
7:    c ← GetClique(G, m, RemainingSet);
8:    if (c ≠ ∅) then
9:       C ← C ∪ {c};
10:   end if
11: end while
12: return C
```

3.5. Local Search Updating

The candidate solution initialized by the initialization procedure is just a good candidate solution meeting the requirement of the DTKWC search problem, but there is no guarantee that it is a great candidate solution. Therefore, in this subsection, we design a local search method to improve the quality of this solution by exploring as many new maximal weight cliques as possible (line 4).

The proposed local search method in Algorithm 4 finds the different maximal weight clique combinations from a candidate solution. It is a good way to iteratively find a better combination $C^{'}$ which includes the vertices with a greater total weight. For this reason, we add a new maximal weight clique $c^{'}$ created by *GetClique* into the current candidate solution C. Then we compute $k+1$ *score* functions explained in Section 3.1 for all of maximal weight cliques in $C^{'}$ each time (line 8). After this, we gain $k+1$ values of *score*, delete the maximal weight clique with the smallest value among these maximal weight cliques in $C^{'}$. Such that we maintain a k size maximal weight clique solution as the new candidate solution (line 11).

Algorithm 4 LocalSearch($G, C, m, RemainingSet$)

```
1: step ← 0, C' ← ∅;
2: while (elapsed time < cutoff) do
3:    step ← step + 1;
4:    c' ← GetClique(G, m, RemainingSet);
5:    if (c' = ∅) then
6:       break;
7:    end if
8:    C' ← C ∪ {c'};
9:    Compute score(c) of each maximal weight clique c in C';
10:   c_min ← arg_{c∈C'} min{score(c)};
11:   Remove c_min from C', breaking ties in favour of the smaller one;
12:   if (cov(C') > cov(C)) then
13:      C ← C', step ← 0;
14:   end if
15:   /* fs is the third parameter of TOPKWCLQ */
16:   if (step ≥ fs) then
17:      break;
18:   end if
19: end while
20: return C
```

Although, we obtain the information that sometimes a candidate solution cannot be improved by the normal local search method in a long time. For this, we add a fixed step,

denoted by fs, into our local search framework that breaks the loop if the current solution cannot be improved in fs steps.

3.6. An Example of DTKWC Search Problem

Example 1. *Let us illustrate how to explore a solution for the DTKWC search problem by using a sample weighted graph in Figure 2.*

Figure 2a gives an weighted graph $G(V, E, w)$ with ten vertices, where $v_i^{w_i}$ denotes vertex v_i with $w_i = w(v_i)$. Assume the integer parameter $k = 2$. The best clique weight so far is $\omega(C_{max}) = 13$. During the first phase, *InitKCliques* creates a maximal weight clique set $C_{max} = \{c_1, c_2\}$, and $c_1 = \{v_0^1, v_1^2, v_2^1, v_3^2\}$, $c_2 = \{v_3^2, v_4^3, v_5^3\}$. The total weight of C_{max} is 12. In the second phase, *LocalSearch* tries to determine a new maximal weight clique c_3 by *GetClique*. Suppose $c_3 = \{v_5^3, v_7^3\}$. We add c_3 into C_{max}. Now, C_{max} contains 3 maximal weight cliques. We evaluate the quality of these three maximal weight cliques by the scoring function we proposed. $\{v_0^1, v_1^2, v_2^1\}$, $\{v_4^3\}$, $\{v_7^3\}$ are the private vertices set of c_1, c_2, c_3, respectively. $score(c_1) = 4$, $score(c_2) = 3$, $score(c_3) = 3$. After this process, we remove the worst maximal weight clique c_2 or c_3 to keep the size of the maximal weight clique set equal to 2. Observe that the set of $\{c_1, c_2\}$ or $\{c_1, c_3\}$ are both the solution of DTKWC search problem in this graph with the parameter $k = 2$, and $\{c_1, c_3\}$ is the best solution with the lowest overlapping.

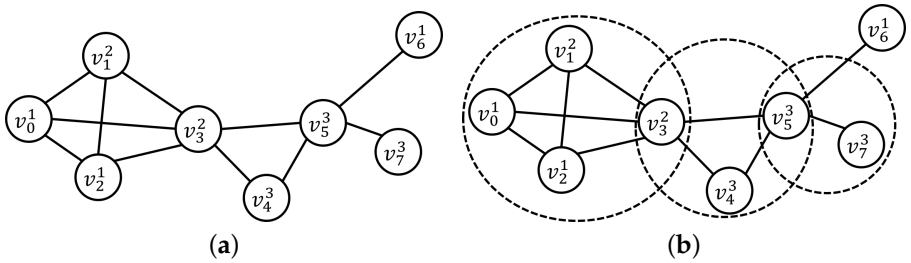

Figure 2. A simple example for DTKWC search problem. (**a**) A weighted graph with 7 vertices. (**b**) Graph with three maximal weight cliques.

4. Experimental Evaluation

In this section, we carry out extensive experiments to evaluate the performance of TOPKWCLQ on weighted real-world large graphs. Since there is no suitable heuristic or exact algorithm for the DTKWC search problem on real-world large graphs in literature, as we know that is a good choice to compare the results of the proposed algorithm to the results obtained by CPLEX solver which is a commercial solver for many combinatorial optimization problems with their constraint formulas of mathematical models. Therefore, the results obtained by CPLEX can be used as reference on the solution quality. We first describe the weighted benchmark and then present the experimental preliminaries and introduce the parameter settings.

4.1. The Benchmark

We evaluate the TOPKWCLQ algorithm on the benchmarks of the weighted real-world graph, which will be shown below.

The weighted real-world large graph benchmark in our experiments was originally from the Network Data Repository online [18] (http://www.graphrepository.com/networks.php, accessed on 1 August 2021). There are millions of vertices and tens of millions of edges on many of the real-world graphs which used in our experiments. This benchmark has been transformed from unweighted graphs to the weighted graphs used the weighting function $w(v_i) = (i \ mod \ 200) + 1$ (including 102 instances) [19]. Moreover, most of these as the experimental instances used in maximum vertex weight clique problem [6,20,21], coloring problem [22], maximum k-plexes problem [23] and DTKC

search problem [2]. Considering the relationship between the DTKWC search problem and these problems, these real-world graphs can naturally be used to evaluate the performance of our algorithm for the DTKWC search problem. These real-world graphs were downloaded from the author's website (http://lcs.ios.ac.cn/~caisw/Resource/weighted-massive-graphs.zip, accessed on 1 August 2021).

The graphs in our experiments are divided into 11 classes, including biological networks, collaboration networks, interaction networks, infrastructure networks, recommendation networks, retweet networks, scientific computing, social networks, facebook networks, technological networks, and web graphs.

4.2. Experimental Preliminaries and Parameter Tuning

The proposed algorithm TOPKWCLQ was implemented in C++ and compiled on CentOS with 2.4 GHz CPU and 32G RAM with "-O3" flag. We run TOPKWCLQ 10 times independently with the random seed setting from 1 to 10 for each instances. Each one is run until the run time of the algorithm arrives which is a given time limit that is assigned as 600 s in this paper. The termination criterion of CPLEX is either the convergence of lower and upper bounds or a time limit which is assigned as 3600 s. We use the solution values of CPLEX to evaluate the quality of the solution solved by TOPKWCLQ.

For each real-world large graph used in our experiments, we set the parameter k to 10, 20, 30, 40, and 50 to obtain five DTKWC search problem instances. Hence, there were $102 \times 5 = 510$ DTKWC search problem instances in our experiments.

TOPKWCLQ uses three parameters for which well-working values must be found: m_0 and m_{max} are the minimum and maximum value of BMS strategy respectively, and fs is the maximum allowed updating steps of the solution per iteration. Parameters m_0 and m_{max} are used in the BMS strategy inspired by [17]. The value of these parameters are set in Table 1 according to a preliminary tuning experiment.

Table 1. Setting of parameters m_0, m_{max}, fs.

Parameters	Descriptions	Range	Values
m_0	minimum number of greediness iterations in BMS strategy	2, 4, 8, 16	8
m_{max}	maximum number of greediness iterations in BMS strategy	32, 64, 128, 256	64
fs	control restart in update procedure	1000, 2000, 4000, 8000	2000

The next subsection is shown to the evaluation of TOPKWCLQ compared with the lower bound ("LB") and the upper bound ("UB") of CPLEX under all 510 DTKWC search problem instances.

4.3. Experimental Results

We present the comprehensive experiment results on the benchmark instances described in Section 4.1 with 5 values of parameter k in Tables 2–6. Among them, Tables 2–6 for $k = 10, 20, 30, 40, 50$, respectively.

For each instance, the column "Instance" indicates the basic information for the name. In TOPKWCLQ, we present the maximum weight value of the DTKWC search problem instances (w_b) and the average weight DTKWC search problem results (w_a) obtained over 10 runs. We also report the average run time over 10 runs (*Time*, in seconds) to reach the maximum weight for all DTKWC search problem instances by TOPKWCLQ. And "0" in the time column indicates TOPKWCLQ was able to obtain the best solution in less than 0.01 s. To study the effectiveness of TOPKWCLQ for DTKWC search problem, we compare it with the CPLEX solver (version 12.9) with the mathematical model (8)–(13) introduced in Section 2.2. The best lower bound (LB) and upper bound (UB) found by CPLEX are listed in the CPLEX columns. If CPLEX was unable to find a bound on an instance, the corresponding

entry is marked by "-". If CPLEX was unable to load the model, the entry is marked by "N/A". For the items in a column, the bold value indicates that the algorithm obtained the same or better objective values compared to the results of the comparison algorithm.

Table 2. Experiment results on real-world large graphs with $k = 10$.

Instance	CPLEX		TOPKWCLQ			Instance	CPLEX		TOPKWCLQ		
	LB	UB	w_b	w_a	$Time$		LB	UB	w_b	w_a	$Time$
bio-celegans	**6867**	**6867**	6852	6808.4	250.97	socfb-B-anon	N/A	N/A	**21,273**	20,893.6	198.57
bio-diseasome	**8672**	**8672**	**8672**	**8672**	0.11	socfb-Berkeley13	N/A	N/A	**38,740**	38,354.7	103.04
bio-dmela	N/A	N/A	**6094**	5994.9	137.32	socfb-CMU	N/A	N/A	**32,888**	32,488.2	143.52
bio-yeast	**4991**	5020	**4991**	**4991**	4.70	socfb-Duke14	N/A	N/A	**30,722**	30,284.8	199.64
ca-AstroPh	N/A	N/A	**47,912**	47,445	152.59	socfb-Indiana	N/A	N/A	**45,026**	44,249.2	155.71
ca-citeseer	N/A	N/A	**76,782**	76,398	244.19	socfb-MIT	N/A	N/A	**31,207**	30,894.1	196.22
ca-coauthors-dblp	N/A	N/A	**304,932**	302,045	177.41	socfb-OR	N/A	N/A	**27,601**	27,282.7	186.37
ca-CondMat	N/A	N/A	**20,070**	19,949.8	187.83	socfb-Penn94	N/A	N/A	**39,164**	38,537	210.69
ca-CSphd	**4059**	**4059**	**4059**	**4059**	0.30	socfb-Stanford3	N/A	N/A	**38,668**	38,137.4	202.05
ca-dblp-2010	N/A	N/A	**64,732**	64,245.7	168.86	socfb-Texas84	N/A	N/A	**43,443**	42,722.4	153.66
ca-dblp-2012	N/A	N/A	**64,179**	64,024.8	209.09	socfb-uci-uni	N/A	N/A	**7092**	6911.8	174.43
ca-Erdos992	N/A	N/A	**5800**	5664.5	215.51	socfb-UCLA	N/A	N/A	**42,919**	42,595	133.01
ca-GrQc	N/A	N/A	**25,844**	25,844	0.16	socfb-UConn	N/A	N/A	**38,261**	38,075.5	214.20
ca-HepPh	N/A	N/A	**79,624**	79,496.8	225.91	socfb-UCSB37	N/A	N/A	**42,830**	42,442.4	166.30
ca-hollywood-2009	N/A	N/A	**91,3861**	90,8334.9	223.68	socfb-UF	N/A	N/A	**52,626**	52,133.5	231.33
ca-MathSciNet	N/A	N/A	**24,017**	23,826	156.59	socfb-UIllinois	N/A	N/A	**46,705**	46,144.6	219.00
ca-netscience	**7588**	**7588**	**7588**	**7588**	0.16	socfb-Wisconsin87	N/A	N/A	**33,594**	33,384.2	183.56
ia-email-EU	N/A	N/A	**8440**	8236.2	239.39	soc-flickr	N/A	N/A	**26,965**	26,228.7	173.67
ia-email-univ	**9570**	9718	9566	9536.2	236.42	soc-flixster	N/A	N/A	**25,505**	24,409.3	158.30
ia-enron-large	N/A	N/A	**16,124**	15,984	194.00	soc-FourSquare	N/A	N/A	**12,254**	11,894.3	180.01
ia-enron-only	**4331**	4338	**4331**	4331	13.55	soc-gowalla	N/A	N/A	**19,607**	19,371.3	174.70
ia-fb-messages	5738	7441	**5792**	5737.7	211.06	soc-karate	**472**	**472**	**472**	**472**	0.00
ia-infect-dublin	**10,946**	11,001	**10,946**	10,946	9.17	soc-lastfm	N/A	N/A	**13,122**	12,833.9	198.74
ia-infect-hyper	**4446**	5292	4442	4416.3	160.16	soc-livejournal	N/A	N/A	**138,283**	135,583.1	145.34
ia-reality	N/A	N/A	**2882**	2852.8	212.69	soc-LiveMocha	N/A	N/A	**9869**	9608.8	148.29
ia-wiki-Talk	N/A	N/A	**10,976**	10,754	172.35	soc-orkut	N/A	N/A	**42,484**	41,987.8	145.63
inf-power	N/A	N/A	**6613**	6613	37.32	soc-pokec	N/A	N/A	**21,995**	21,094.4	130.38
inf-roadNet-CA	N/A	N/A	**6367**	6296.1	141.26	soc-slashdot	N/A	N/A	**13,801**	13,224.5	169.68
inf-roadNet-PA	N/A	N/A	**6084**	6082.5	148.54	soc-twitter-follows	N/A	N/A	**5535**	5346.8	174.89
inf-road-usa	N/A	N/A	**6035**	6002.5	136.92	soc-wiki-Vote	**6376**	6768	6341	6253.2	133.06
rec-amazon	N/A	N/A	**8931**	8931	1.73	soc-youtube	N/A	N/A	**12,409**	12,255	191.54
rt-retweet	**1575**	1578	**1575**	**1575**	0.06	soc-youtube-snap	N/A	N/A	**12,146**	11,973.7	140.26
rt-retweet-crawl	N/A	N/A	**8742**	8650.5	189.03	tech-as-caida2007	N/A	N/A	**7240**	7111	196.61
rt-twitter-copen	**4661**	**4661**	**4661**	4659.5	128.18	tech-as-skitter	N/A	N/A	**29,470**	28,331.9	131.52
sc-ldoor	N/A	N/A	**40,726**	40,704.3	179.99	tech-internet-as	N/A	N/A	**7457**	7067.2	166.47
sc-msdoor	N/A	N/A	**40,670**	40,625.9	135.87	tech-p2p-gnutella	N/A	N/A	**5829**	5816.2	190.24
sc-nasasrb	N/A	N/A	**43,776**	43,714	173.28	tech-RL-caida	N/A	N/A	**12,215**	11,994.8	164.86
sc-pkustk11	N/A	N/A	**47,548**	47,268.1	153.85	tech-routers-rf	3914	207,554	**10,002**	9871	128.18
sc-pkustk13	N/A	N/A	**57,282**	56,738	185.36	tech-WHOIS	N/A	N/A	**31,459**	31,125.1	144.75
sc-pwtk	N/A	N/A	**45,504**	45,432	179.58	web-arabic-2005	N/A	N/A	**92,108**	**92,108**	3.98
sc-shipsec1	N/A	N/A	**31,790**	31,661.2	214.63	web-BerkStan	N/A	N/A	**11,200**	**11,200**	71.12
sc-shipsec5	N/A	N/A	**43,260**	43,157.4	132.02	web-edu	N/A	N/A	**11,250**	**11,250**	0.17
soc-BlogCatalog	N/A	N/A	**20,014**	19,109.1	242.59	web-google	**13,126**	**13,126**	**13,126**	**13,126**	0.01
soc-brightkite	N/A	N/A	**19,548**	19,178.3	173.03	web-indochina-2004	N/A	N/A	**44,052**	**44,052**	44.42
soc-buzznet	N/A	N/A	**17,620**	17,041.3	199.35	web-it-2004	N/A	N/A	**415,850**	**415,850**	0.63
soc-delicious	N/A	N/A	**11,711**	11,553.1	136.82	web-polblogs	**5926**	6428	5918	5828.2	166.71
soc-digg	N/A	N/A	**27,726**	27,083.9	226.17	web-sk-2005	N/A	N/A	**63,930**	63,920.4	132.89
soc-dolphins	**1226**	**1226**	**1226**	**1226**	0.04	web-spam	N/A	N/A	**14,569**	14,418.8	150.91
soc-douban	N/A	N/A	**8983**	8860.1	207.39	web-uk-2005	N/A	N/A	**441,613**	**441,613**	0.41
soc-epinions	N/A	N/A	**12,942**	12,661.1	139.36	web-webbase-2001	N/A	N/A	**20,648**	**20,648**	68.16
socfb-A-anon	N/A	N/A	**22,551**	21,896.8	176.24	web-wikipedia2009	N/A	N/A	**29,781**	29,063.8	162.08

Table 3. Experiment results on real-world large graphs with $k = 20$.

Instance	CPLEX		TOPKWCLQ			Instance	CPLEX		TOPKWCLQ		
	LB	*UB*	w_b	w_a	*Time*		*LB*	*UB*	w_b	w_a	*Time*
bio-celegans	**11,489**	11,747	11,275	11,198.2	147.50	socfb-B-anon	N/A	N/A	**38,667**	38,284.5	221.32
bio-diseasome	**14,313**	14,345	**14,313**	**14,313**	2.06	socfb-Berkeley13	N/A	N/A	**67,518**	66,896.4	158.57
bio-dmela	N/A	N/A	**10,447**	10,370.9	209.09	socfb-CMU	N/A	N/A	**55,267**	54,811	191.40
bio-yeast	8874	9730	**9067**	9060	125.32	socfb-Duke14	N/A	N/A	**54,844**	54,091.1	137.00
ca-AstroPh	N/A	N/A	**85,366**	85,173.6	185.29	socfb-Indiana	N/A	N/A	**80,040**	79,382.1	162.68
ca-citeseer	N/A	N/A	**131,947**	131,244.5	148.71	socfb-MIT	N/A	N/A	**55,086**	54,421.2	156.99
ca-coauthors-dblp	N/A	N/A	**531,666**	529,145	235.05	socfb-OR	N/A	N/A	**48,737**	48,162.2	216.69
ca-CondMat	N/A	N/A	**35,029**	34,734	249.30	socfb-Penn94	N/A	N/A	**70,888**	70,417.4	178.75
ca-CSphd	-	-	**7943**	7942	103.99	socfb-Stanford3	N/A	N/A	**66,445**	65,926.7	213.33
ca-dblp-2010	N/A	N/A	**107,838**	107,254.7	192.43	socfb-Texas84	N/A	N/A	**75,522**	74,283.2	156.21
ca-dblp-2012	N/A	N/A	**102,895**	102,577.8	214.70	socfb-uci-uni	N/A	N/A	**13,586**	13,267.4	191.29
ca-Erdos992	N/A	N/A	**9766**	9580.7	146.01	socfb-UCLA	N/A	N/A	**72,363**	71,465.5	192.25
ca-GrQc	N/A	N/A	**37,559**	37,559	24.67	socfb-UConn	N/A	N/A	**64,266**	63,327	202.52
ca-HepPh	N/A	N/A	**113,984**	113,639.5	189.33	socfb-UCSB37	N/A	N/A	**68,722**	68,234.3	135.47
ca-hollywood-2009	N/A	N/A	**1,305,029**	1,293,108	185.10	socfb-UF	N/A	N/A	**90,307**	89,178.6	233.87
ca-MathSciNet	N/A	N/A	**41,461**	41,215.4	137.01	socfb-UIllinois	N/A	N/A	**79,952**	79,253.7	243.79
ca-netscience	13,178	13,196	**13,189**	**13,189**	2.16	socfb-Wisconsin87	N/A	N/A	**59,743**	59,398.8	215.07
ia-email-EU	N/A	N/A	**13,440**	13,232.2	151.90	soc-flickr	N/A	N/A	**44,279**	43,612.1	194.81
ia-email-univ	11,024	18,426	**15,647**	15,525.2	172.17	soc-flixster	N/A	N/A	**41,715**	40,413	210.02
ia-enron-large	N/A	N/A	**28,592**	28,252.6	123.99	soc-FourSquare	N/A	N/A	**20,866**	20,602.4	213.53
ia-enron-only	6922	7237	**6923**	6917	177.34	soc-gowalla	N/A	N/A	**35,017**	34,396	185.05
ia-fb-messages	6204	15,510	**10,305**	10,207.4	189.05	soc-karate	**629**	**629**	**629**	**629**	0.00
ia-infect-dublin	15,009	18,694	**17,588**	17,527.2	215.34	soc-lastfm	N/A	N/A	**23,753**	22,918.3	113.19
ia-infect-hyper	5972	6554	**6086**	6068.2	116.70	soc-livejournal	N/A	N/A	**222,938**	220,213.4	147.43
ia-reality	N/A	N/A	**5291**	5252.3	161.50	soc-LiveMocha	N/A	N/A	**16,907**	16,391.4	106.99
ia-wiki-Talk	N/A	N/A	**18,644**	18,315.4	252.87	soc-orkut	N/A	N/A	**79,351**	78,363.4	150.28
inf-power	N/A	N/A	**11,743**	11,712	204.17	soc-pokec	N/A	N/A	**38,793**	38,128.1	239.65
inf-roadNet-CA	N/A	N/A	**12,341**	12,292.8	179.86	soc-slashdot	N/A	N/A	**21,071**	20,868.8	217.39
inf-roadNet-PA	N/A	N/A	**12,049**	12,041.1	157.62	soc-twitter-follows	N/A	N/A	**9722**	9588.3	208.03
inf-road-usa	N/A	N/A	**11,940**	11,827.3	191.68	soc-wiki-Vote	10,649	12,613	**10,670**	10,620.8	225.30
rec-amazon	N/A	N/A	**17,414**	**17,414**	10.02	soc-youtube	N/A	N/A	**22,173**	21,901.9	199.76
rt-retweet	2732	2758	**2754**	**2754**	19.86	soc-youtube-snap	N/A	N/A	**21,687**	21,423.5	162.58
rt-retweet-crawl	N/A	N/A	**15,283**	14,983.6	163.67	tech-as-caida2007	N/A	N/A	**11,632**	11,485.1	114.11
rt-twitter-copen	**8364**	8525	8293	8256.7	142.45	tech-as-skitter	N/A	N/A	**47,288**	44,902.5	169.13
sc-ldoor	N/A	N/A	**81,165**	81,074.7	202.88	tech-internet-as	N/A	N/A	**12,591**	11,964.9	157.32
sc-msdoor	N/A	N/A	**80,703**	80,651.2	178.45	tech-p2p-gnutella	N/A	N/A	**11,115**	11,034.6	168.48
sc-nasasrb	N/A	N/A	**84,638**	84,212.2	162.43	tech-RL-caida	N/A	N/A	**21,048**	20,652.7	83.62
sc-pkustk11	N/A	N/A	**90,376**	90,205.5	204.08	tech-routers-rf	N/A	N/A	**15,456**	15,301.5	173.66
sc-pkustk13	N/A	N/A	**108,096**	107,515.5	141.38	tech-WHOIS	N/A	N/A	**44,189**	43,782.9	188.40
sc-pwtk	N/A	N/A	**89,496**	89,366	204.28	web-arabic-2005	N/A	N/A	**178,434**	**178,434**	3.22
sc-shipsec1	N/A	N/A	**60,831**	60,323.5	156.03	web-BerkStan	N/A	N/A	**18,756**	18,724	143.16
sc-shipsec5	N/A	N/A	**82,422**	81,832.3	133.56	web-edu	N/A	N/A	**16,531**	16,498	170.07
soc-BlogCatalog	N/A	N/A	**30,479**	29,811	201.55	web-google	14,174	22,943	**21,479**	**21,479**	0.59
soc-brightkite	N/A	N/A	**31,062**	30,674.7	207.00	web-indochina-2004	N/A	N/A	**74,320**	74,236.2	218.99
soc-buzznet	N/A	N/A	**28,642**	28,133.5	193.13	web-it-2004	N/A	N/A	**797,123**	**797,123**	1.03
soc-delicious	N/A	N/A	**20,402**	20,228.4	92.03	web-polblogs	9808	11,906	**9949**	9909.4	135.08
soc-digg	N/A	N/A	**42,903**	42,169.9	170.04	web-sk-2005	N/A	N/A	**97,196**	97,053	192.90
soc-dolphins	**1861**	1871	**1861**	**1861**	6.05	web-spam	N/A	N/A	**23,982**	23,625.7	175.29
soc-douban	N/A	N/A	**15,043**	14,670.5	198.98	web-uk-2005	N/A	N/A	**789,896**	789,896	0.43
soc-epinions	N/A	N/A	**22,001**	21,671.5	247.93	web-webbase-2001	N/A	N/A	**33,438**	33,290.2	154.66
socfb-A-anon	N/A	N/A	**39,458**	39,201.1	210.51	web-wikipedia2009	N/A	N/A	**48,222**	47,165.3	230.13

Table 4. Experiment results on real-world large graphs with $k = 30$.

Instance	CPLEX		TOPKWCLQ			Instance	CPLEX		TOPKWCLQ		
	LB	*UB*	w_b	w_a	*Time*		*LB*	*UB*	w_b	w_a	*Time*
bio-celegans	14,528	16,193	**14,805**	14,740	203.67	socfb-B-anon	N/A	N/A	**55,318**	54,993.8	155.81
bio-diseasome	12,610	20,103	**18,431**	18,425.5	91.62	socfb-Berkeley13	N/A	N/A	**91,991**	90,393.9	162.54
bio-dmela	N/A	N/A	**14,497**	14,384	180.80	socfb-CMU	N/A	N/A	**72,885**	72,032.9	162.68
bio-yeast	-	-	**12,774**	12,750.3	198.29	socfb-Duke14	N/A	N/A	**75,543**	74,516.6	149.19
ca-AstroPh	N/A	N/A	**117,328**	116,221.3	241.64	socfb-Indiana	N/A	N/A	**110,454**	109,407.1	198.30
ca-citeseer	N/A	N/A	**178,715**	177,726.9	209.84	socfb-MIT	N/A	N/A	**74,158**	73,595.9	182.28
ca-coauthors-dblp	N/A	N/A	**740,505**	733,645.3	135.01	socfb-OR	N/A	N/A	**67,317**	66,836.6	179.21
ca-CondMat	N/A	N/A	**48,476**	48,219.6	138.60	socfb-Penn94	N/A	N/A	**98,255**	97,120.6	165.00
ca-CSphd	N/A	N/A	**11,739**	11,731	150.83	socfb-Stanford3	N/A	N/A	**90,442**	89,784.3	151.78
ca-dblp-2010	N/A	N/A	**146,421**	145,317.6	171.76	socfb-Texas84	N/A	N/A	**103,092**	101,991	212.90
ca-dblp-2012	N/A	N/A	**133,948**	133,150.3	106.77	socfb-uci-uni	N/A	N/A	**19,436**	19,111	173.62
ca-Erdos992	N/A	N/A	**13,381**	13,305.1	173.98	socfb-UCLA	N/A	N/A	**97,506**	96,350.8	206.58
ca-GrQc	N/A	N/A	**46,262**	46,161.4	151.01	socfb-UConn	N/A	N/A	**84,391**	83,777.7	183.23
ca-HepPh	N/A	N/A	**140,485**	140,068.4	169.58	socfb-UCSB37	N/A	N/A	**90,198**	89,200.8	181.14
ca-hollywood-2009	N/A	N/A	**1,591,287**	1,578,513	203.39	socfb-UF	N/A	N/A	**119,700**	117,876.9	124.65
ca-MathSciNet	N/A	N/A	**56,826**	56,490.5	202.42	socfb-UIllinois	N/A	N/A	**108,948**	108,470.3	135.60
ca-netscience	17,341	17,997	**17,781**	17,780.1	121.27	socfb-Wisconsin87	N/A	N/A	**83,837**	82,596.5	211.36
ia-email-EU	N/A	N/A	**17,861**	17,546.7	182.28	soc-flickr	N/A	N/A	**59,976**	59,342.1	125.18
ia-email-univ	8733	26,220	**20,749**	20,538.9	201.94	soc-flixster	N/A	N/A	**55,676**	54,557.7	181.08
ia-enron-large	N/A	N/A	**40,233**	39,553.5	189.25	soc-FourSquare	N/A	N/A	**28,775**	28,598.1	233.85
ia-enron-only	8030	9226	**8570**	8542.2	160.40	soc-gowalla	N/A	N/A	**48,365**	47,895.4	148.82
ia-fb-messages	7135	23,605	**14,396**	14,223	152.57	soc-karate	**629**	**629**	**629**	**629**	0.00
ia-infect-dublin	12,608	26,004	**22,783**	22,627.5	193.70	soc-lastfm	N/A	N/A	**32,614**	32,036.3	157.81
ia-infect-hyper	**6554**	**6554**	**6554**	**6554**	1.68	soc-livejournal	N/A	N/A	**297,475**	289,923.2	167.77
ia-reality	N/A	N/A	**7473**	7404.8	205.85	soc-LiveMocha	N/A	N/A	**23,056**	22,702.8	187.07
ia-wiki-Talk	N/A	N/A	**25,589**	25,136.2	172.33	soc-orkut	N/A	N/A	**114,442**	112,839.3	187.46
inf-power	N/A	N/A	**16,414**	16,356.8	172.16	soc-pokec	N/A	N/A	**55,396**	54,045.9	170.75
inf-roadNet-CA	N/A	N/A	**18,351**	18,245.3	232.32	soc-slashdot	N/A	N/A	**28,032**	27,717.9	261.61
inf-roadNet-PA	N/A	N/A	**18,005**	17,990.4	186.22	soc-twitter-follows	N/A	N/A	**13,690**	13,527.6	203.29
inf-road-usa	N/A	N/A	**17,728**	17,649.3	156.50	soc-wiki-Vote	9944	18,426	**14,496**	14,390.8	248.59
rec-amazon	N/A	N/A	**25,468**	**25,468**	112.68	soc-youtube	N/A	N/A	**30,905**	30,491.7	185.84
rt-retweet	**3609**	**3609**	**3609**	**3609**	21.53	soc-youtube-snap	N/A	N/A	**30,746**	30,309.2	203.55
rt-retweet-crawl	N/A	N/A	**21,151**	20,857.2	230.71	tech-as-caida2007	N/A	N/A	**16,120**	15,863.9	138.62
rt-twitter-copen	**11,777**	12,474	11,590	11,548.8	161.99	tech-as-skitter	N/A	N/A	**60,308**	58,536.9	151.54
sc-ldoor	N/A	N/A	**121,303**	121,165.8	214.11	tech-internet-as	N/A	N/A	**16,818**	16,538	131.74
sc-msdoor	N/A	N/A	**120,426**	120,330.8	250.58	tech-p2p-gnutella	N/A	N/A	**16,023**	15,920.1	239.06
sc-nasasrb	N/A	N/A	**123,112**	122,866	197.63	tech-RL-caida	N/A	N/A	**28,826**	28,485.9	174.90
sc-pkustk11	N/A	N/A	**132,640**	132,153	244.31	tech-routers-rf	N/A	N/A	**20,062**	19,967.9	145.42
sc-pkustk13	N/A	N/A	**157,978**	156,744.2	221.26	tech-WHOIS	N/A	N/A	**53,750**	53,301.1	159.00
sc-pwtk	N/A	N/A	**132,744**	132,648.4	245.68	web-arabic-2005	N/A	N/A	**263,602**	**263,602**	7.09
sc-shipsec1	N/A	N/A	**87,853**	87,048.3	150.32	web-BerkStan	N/A	N/A	**25,848**	25,786.1	122.40
sc-shipsec5	N/A	N/A	**118,524**	117,849.5	174.37	web-edu	N/A	N/A	**20,805**	20,751	198.34
soc-BlogCatalog	N/A	N/A	**40,724**	39,594.4	184.04	web-google	10,767	28,599	**26,866**	26,858.8	131.36
soc-brightkite	N/A	N/A	**41,189**	40,361.4	212.23	web-indochina-2004	N/A	N/A	**98,809**	98,657.3	176.93
soc-buzznet	N/A	N/A	**38,628**	37,701	174.26	web-it-2004	N/A	N/A	**1,115,823**	**1,115,823**	1.32
soc-delicious	N/A	N/A	**28,263**	27,792.6	135.52	web-polblogs	8579	17,312	**13,549**	13,498.1	171.71
soc-digg	N/A	N/A	**55,879**	54,964.3	164.61	web-sk-2005	N/A	N/A	**126,823**	126,524.1	144.92
soc-dolphins	**2015**	**2015**	**2015**	**2015**	0.00	web-spam	N/A	N/A	**31,803**	31,327.7	134.99
soc-douban	N/A	N/A	**20,017**	19,862	201.77	web-uk-2005	N/A	N/A	**1,050,477**	**105,0477**	0.44
soc-epinions	N/A	N/A	**29,866**	29,612.2	209.61	web-webbase-2001	N/A	N/A	**44,256**	43,708.3	156.66
socfb-A-anon	N/A	N/A	**57,408**	55,907.1	157.96	web-wikipedia2009	N/A	N/A	**65,286**	63,121.4	198.87

Table 5. Experiment results on real-world large graphs with $k = 40$.

Instance	CPLEX		TOPKWCLQ			Instance	CPLEX		TOPKWCLQ		
	LB	*UB*	w_b	w_a	*Time*		*LB*	*UB*	w_b	w_a	*Time*
bio-celegans	10,037	22,077	**18,057**	17,956.7	218.25	socfb-B-anon	N/A	N/A	**71,895**	70,879.8	170.92
bio-diseasome	10,450	24,138	**21,945**	21,933.6	188.25	socfb-Berkeley13	N/A	N/A	**113,862**	111,784.4	173.56
bio-dmela	N/A	N/A	**18,403**	18,266.6	197.28	socfb-CMU	N/A	N/A	**87,972**	86,992.2	180.33
bio-yeast	-	-	**16,302**	16,247.8	182.60	socfb-Duke14	N/A	N/A	**93,294**	92,967.7	224.83
ca-AstroPh	N/A	N/A	**144,660**	143,098.9	186.93	socfb-Indiana	N/A	N/A	**135,869**	135,265	178.45
ca-citeseer	N/A	N/A	**218,277**	217,906.1	194.61	socfb-MIT	N/A	N/A	**90,760**	89,700.5	182.88
ca-coauthors-dblp	N/A	N/A	**924,023**	921,156.3	170.24	socfb-OR	N/A	N/A	**85,045**	84,407.4	133.30
ca-CondMat	N/A	N/A	**61,647**	60,964.7	221.77	socfb-Penn94	N/A	N/A	**121,319**	120,455.7	179.44
ca-CSphd	N/A	N/A	**15,398**	15,370.2	206.08	socfb-Stanford3	N/A	N/A	**112,011**	111,138.3	140.08
ca-dblp-2010	N/A	N/A	**180,532**	179,364.5	182.20	socfb-Texas84	N/A	N/A	**128,204**	126,768.1	219.27
ca-dblp-2012	N/A	N/A	**160,660**	159,929.8	169.46	socfb-uci-uni	N/A	N/A	**25,650**	25,101.6	226.36
ca-Erdos992	N/A	N/A	**16,877**	16,752.9	125.53	socfb-UCLA	N/A	N/A	**119,747**	118,738.3	187.82
ca-GrQc	N/A	N/A	**53,645**	53,532.1	170.29	socfb-UConn	N/A	N/A	**102,092**	101,497.7	207.71
ca-HepPh	N/A	N/A	**161,167**	160,718.8	207.33	socfb-UCSB37	N/A	N/A	**108,210**	107,448.6	210.05
ca-hollywood-2009	N/A	N/A	**1,815,958**	1,803,757	213.10	socfb-UF	N/A	N/A	**143,528**	142,751.4	126.65
ca-MathSciNet	N/A	N/A	**70,636**	70,271.1	143.10	socfb-UIllinois	N/A	N/A	**135,222**	134,182.4	201.21
ca-netscience	10,519	22,011	**21,076**	21,072.8	136.11	socfb-Wisconsin87	N/A	N/A	**104,664**	103,275.1	203.96
ia-email-EU	N/A	N/A	**21,652**	21,347.9	217.56	soc-flickr	N/A	N/A	**74,830**	74,224.6	160.25
ia-email-univ	10,977	32,641	**25,202**	25,079.5	174.42	soc-flixster	N/A	N/A	**68,593**	67,034.3	182.67
ia-enron-large	N/A	N/A	**50,536**	50,100.5	197.78	soc-FourSquare	N/A	N/A	**36,905**	36,496.1	180.98
ia-enron-only	8159	10,121	**9610**	9590.5	123.20	soc-gowalla	N/A	N/A	**61,166**	60,477.1	106.86
ia-fb-messages	-	-	**18,078**	17,939.4	155.67	soc-karate	**629**	**629**	**629**	**629**	0.00
ia-infect-dublin	17,535	30,618	**26,721**	26,652.9	207.08	soc-lastfm	N/A	N/A	**40,830**	40,624.8	191.26
ia-infect-hyper	**6554**	**6554**	**6554**	**6554**	0.00	soc-livejournal	N/A	N/A	**353,252**	345,997.4	159.91
ia-reality	N/A	N/A	**9474**	9427.1	159.85	soc-LiveMocha	N/A	N/A	**29,376**	28,832	190.46
ia-wiki-Talk	N/A	N/A	**31,492**	31,299.6	216.93	soc-orkut	N/A	N/A	**146,600**	145,529.7	135.93
inf-power	N/A	N/A	**20,842**	20,757.7	129.00	soc-pokec	N/A	N/A	**70,454**	69,295.1	140.46
inf-roadNet-CA	N/A	N/A	**24,261**	24,204.7	150.68	soc-slashdot	N/A	N/A	**34,450**	33,981	205.31
inf-roadNet-PA	N/A	N/A	**23,940**	23,929.1	134.36	soc-twitter-follows	N/A	N/A	**17,550**	17,336.8	220.63
inf-road-usa	N/A	N/A	**23,594**	23,458.5	146.11	soc-wiki-Vote	8107	23,864	**18,068**	17,883.6	175.77
rec-amazon	N/A	N/A	**33,363**	33,356.2	161.28	soc-youtube	N/A	N/A	**39,556**	38,949	91.83
rt-retweet	**4161**	4169	**4161**	**4161**	10.27	soc-youtube-snap	N/A	N/A	**38,929**	38,679.1	195.84
rt-retweet-crawl	N/A	N/A	**27,011**	26,594	138.17	tech-as-caida2007	N/A	N/A	**19,973**	19,779.8	135.87
rt-twitter-copen	**14,772**	16,130	14,657	14,635.9	175.68	tech-as-skitter	N/A	N/A	**72,571**	71,620	209.40
sc-ldoor	N/A	N/A	**161,126**	160,998.6	178.39	tech-internet-as	N/A	N/A	**21,029**	20,875.3	175.95
sc-msdoor	N/A	N/A	**159,967**	159,737.1	224.65	tech-p2p-gnutella	N/A	N/A	**20,773**	20,588	110.00
sc-nasasrb	N/A	N/A	**160,746**	160,572.2	148.46	tech-RL-caida	N/A	N/A	**36,579**	36,137.1	133.97
sc-pkustk11	N/A	N/A	**174,249**	173,122.1	163.16	tech-routers-rf	N/A	N/A	**24,391**	24,223	156.74
sc-pkustk13	N/A	N/A	**204,876**	204,369.1	137.80	tech-WHOIS	N/A	N/A	**61,598**	60,786.6	181.99
sc-pwtk	N/A	N/A	**175,800**	175,532	183.57	web-arabic-2005	N/A	N/A	**348,027**	**348,027**	5.92
sc-shipsec1	N/A	N/A	**113,485**	112,618.6	214.02	web-BerkStan	N/A	N/A	**32,566**	32,486.7	199.25
sc-shipsec5	N/A	N/A	**153,006**	151,986	162.57	web-edu	N/A	N/A	**24,780**	24,730.8	219.82
soc-BlogCatalog	N/A	N/A	**49,422**	48,620.8	193.41	web-google	-	-	**31,201**	31,141.2	148.58
soc-brightkite	N/A	N/A	**49,753**	49,285.4	204.41	web-indochina-2004	N/A	N/A	**119,525**	119,336.2	220.28
soc-buzznet	N/A	N/A	**47,092**	46,465.3	230.39	web-it-2004	N/A	N/A	**1,314,144**	**1,314,144**	9.23
soc-delicious	N/A	N/A	**35,377**	34,841.4	161.33	web-polblogs	9606	21,840	**16,844**	16,756.7	201.77
soc-digg	N/A	N/A	**67,806**	66,826.6	248.28	web-sk-2005	N/A	N/A	**154,004**	153,682.4	175.56
soc-dolphins	**2015**	**2015**	**2015**	**2015**	0.00	web-spam	N/A	N/A	**38,112**	37,909.5	198.68
soc-douban	N/A	N/A	**25,245**	24,822.4	118.98	web-uk-2005	N/A	N/A	**1,277,887**	**1,277,887**	0.45
soc-epinions	N/A	N/A	**37,345**	36,766.1	187.05	web-webbase-2001	N/A	N/A	**52,395**	52,146	192.70
socfb-A-anon	N/A	N/A	**71,908**	71,376.4	198.96	web-wikipedia2009	N/A	N/A	**79,081**	77,242.5	215.47

Table 6. Experiment results on real-world large graphs with $k = 50$.

Instance	CPLEX		TOPKWCLQ			Instance	CPLEX		TOPKWCLQ		
	LB	*UB*	w_b	w_a	*Time*		*LB*	*UB*	w_b	w_a	*Time*
bio-celegans	14,811	25,659	**21,032**	20,899.3	242.15	socfb-B-anon	N/A	N/A	**87,455**	85,964.7	168.6
bio-diseasome	13,041	27,569	**25,157**	25,132.1	219.96	socfb-Berkeley13	N/A	N/A	**132,576**	131,226.4	245.12
bio-dmela	N/A	N/A	**22,060**	21,985	156.03	socfb-CMU	N/A	N/A	**101,002**	100,407.2	224.22
bio-yeast	N/A	N/A	**19,700**	19,584.3	130.19	socfb-Duke14	N/A	N/A	**110,636**	109,983.7	198.54
ca-AstroPh	N/A	N/A	**167,224**	166,691.2	161.95	socfb-Indiana	N/A	N/A	**158,910**	157,463.4	175.19
ca-citeseer	N/A	N/A	**258,664**	256,459.1	200.09	socfb-MIT	N/A	N/A	**104,412**	10,3962	155.76
ca-coauthors-dblp	N/A	N/A	**1,103,639**	1,100,063	173.91	socfb-OR	N/A	N/A	**101,953**	100,836	164.1
ca-CondMat	N/A	N/A	**73,224**	72,874.1	174.02	socfb-Penn94	N/A	N/A	**143,535**	141,653.6	171.98
ca-CSphd	N/A	N/A	**18,913**	18,888.6	215.74	socfb-Stanford3	N/A	N/A	**130,978**	130,307.6	160.01
ca-dblp-2010	N/A	N/A	**211,258**	210,658.3	202.01	socfb-Texas84	N/A	N/A	**150,919**	149,626.2	215.77
ca-dblp-2012	N/A	N/A	**185,410**	184,644.4	138.06	socfb-uci-uni	N/A	N/A	**31,019**	30,807.1	192.24
ca-Erdos992	N/A	N/A	**20,331**	20,133.4	134.71	socfb-UCLA	N/A	N/A	**141,024**	139,929.2	165.78
ca-GrQc	N/A	N/A	**60,410**	60,254	174.04	socfb-UConn	N/A	N/A	**119,269**	118,552.1	183.93
ca-HepPh	N/A	N/A	**177,770**	177,104.2	157.61	socfb-UCSB37	N/A	N/A	**125,680**	124,864.5	169.75
ca-hollywood-2009	N/A	N/A	**2,017,283**	2,004,122	220.66	socfb-UF	N/A	N/A	**166,765**	165,868.4	242.62
ca-MathSciNet	N/A	N/A	**84,035**	83,379.4	168.59	socfb-UIllinois	N/A	N/A	**159,630**	157,918	184.39
ca-netscience	14,917	25,103	**23,696**	23,684.2	146.82	socfb-Wisconsin87	N/A	N/A	**122,934**	122,265.3	172.9
ia-email-EU	N/A	N/A	**25,603**	25,175.5	229.9	soc-flickr	N/A	N/A	**89,807**	87,883	175.89
ia-email-univ	-	-	**29,402**	29,187.6	231.77	soc-flixster	N/A	N/A	**80,451**	79,254.8	212.02
ia-enron-large	N/A	N/A	**60,880**	60,270.8	212.19	soc-FourSquare	N/A	N/A	**44,676**	44,272	194.57
ia-enron-only	9157	10,428	**10,297**	10,266	141.51	soc-gowalla	N/A	N/A	**72,608**	72,149.7	213.64
ia-fb-messages	-	-	**21,654**	21,446	135.39	soc-karate	**629**	**629**	**629**	**629**	0
ia-infect-dublin	21,426	34,156	**30,074**	29,964.5	221.4	soc-lastfm	N/A	N/A	**49,431**	48,941.3	182.21
ia-infect-hyper	**6554**	**6554**	**6554**	**6554**	0	soc-livejournal	N/A	N/A	**405,317**	399,721.7	142.94
ia-reality	N/A	N/A	**11,518**	11,445.6	146.6	soc-LiveMocha	N/A	N/A	**34,890**	34,574.5	194.75
ia-wiki-Talk	N/A	N/A	**38,239**	37,417.3	192.79	soc-orkut	N/A	N/A	**18,1096**	178,069.6	139.86
inf-power	N/A	N/A	**24,968**	24,890.9	224.22	soc-pokec	N/A	N/A	**85,010**	84,118.3	188.24
inf-roadNet-CA	N/A	N/A	**30,251**	30,155.4	111.57	soc-slashdot	N/A	N/A	**40,150**	39,867.5	207.98
inf-roadNet-PA	N/A	N/A	**29,887**	29,867.4	199.69	soc-twitter-follows	N/A	N/A	**21,301**	21,065.8	187.84
inf-road-usa	N/A	N/A	**29,316**	29,201.1	142.64	soc-wiki-Vote	10,097	28,039	**21,419**	21,210.6	187.91
rec-amazon	N/A	N/A	**41,146**	41,124.2	148.14	soc-youtube	N/A	N/A	**47,160**	46,973	183.03
rt-retweet	**4526**	4620	**4526**	**4526**	0.14	soc-youtube-snap	N/A	N/A	**47,523**	46,964.5	234.3
rt-retweet-crawl	N/A	N/A	**32,586**	32,128.1	175.81	tech-as-caida2007	N/A	N/A	**23,881**	23,719.3	164.61
rt-twitter-copen	17,314	19,812	**17,652**	17,571	210.34	tech-as-skitter	N/A	N/A	**84,746**	83,760	193.48
sc-ldoor	N/A	N/A	**200,767**	200,601.1	197.73	tech-internet-as	N/A	N/A	**25,236**	25,078.6	252.39
sc-msdoor	N/A	N/A	**199,076**	198,801.3	129.64	tech-p2p-gnutella	N/A	N/A	**25,238**	25,143.9	187.28
sc-nasasrb	N/A	N/A	**197,922**	197,611.2	186.05	tech-RL-caida	N/A	N/A	**43,728**	43,471.3	151.62
sc-pkustk11	N/A	N/A	**214,700**	213,740.4	140.39	tech-routers-rf	N/A	N/A	**28,349**	28,125.2	137.27
sc-pkustk13	N/A	N/A	**252,454**	251,525.5	186.96	tech-WHOIS	N/A	N/A	**68,102**	67,397.8	165.01
sc-pwtk	N/A	N/A	**218,360**	217,983.9	161.86	web-arabic-2005	N/A	N/A	**430,893**	430,893	12.32
sc-shipsec1	N/A	N/A	**138,034**	137,296.4	181.85	web-BerkStan	N/A	N/A	**39,160**	38,997.3	183.66
sc-shipsec5	N/A	N/A	**185,496**	184,589.5	171.17	web-edu	N/A	N/A	**28,721**	28,667.3	205.28
soc-BlogCatalog	N/A	N/A	**57,097**	56,401.7	150.99	web-google	-	-	**34,977**	34,930.2	194.03
soc-brightkite	N/A	N/A	**58,014**	57,698.3	213.89	web-indochina-2004	N/A	N/A	**138,107**	137,833.2	161.98
soc-buzznet	N/A	N/A	**55,420**	54,720	209.46	web-it-2004	N/A	N/A	**1,502,580**	1,502,580	31.06
soc-delicious	N/A	N/A	**42,317**	41,594	169.03	web-polblogs	9793	25,434	**19,890**	19,727.5	208.03
soc-digg	N/A	N/A	**79,149**	77,969.7	168.98	web-sk-2005	N/A	N/A	**180,230**	179,935.1	183.74
soc-dolphins	**2015**	**2015**	**2015**	**2015**	0	web-spam	N/A	N/A	**45,228**	44,432.1	172.25
soc-douban	N/A	N/A	**29,993**	29,587.2	135.61	web-uk-2005	N/A	N/A	**1,497,314**	**1,497,314**	0.46
soc-epinions	N/A	N/A	**43,676**	43,427.1	179.86	web-webbase-2001	N/A	N/A	**59,569**	59,108.5	153.63
socfb-A-anon	N/A	N/A	**87,217**	86,621.5	203.76	web-wikipedia2009	N/A	N/A	**90,898**	90,000.9	190.59

Tables 2–6 show that TOPKWCLQ obtained the same or better objective values compared with the objective values of CPLEX on most instances. On 5 out of 102 graphs, CPLEX can find better objective values than our algorithm TOPKWCLQ. However, the instances become more challenging for CPLEX with a larger k, and TOPKWCLQ is becoming more

effective. For 19 out of 510 real-world instances, we can prove the optimal solutions, where the values of the lower bound ("LB") and upper bound ("UB") are equivalent in CPLEX column. In terms of computational time, TOPKWCLQ can obtain the optimal values in less than one second (at most hundreds of seconds) in most cases. For example, on the graph rt-twitter-copen, CPLEX always finds better objective values than TOPKWCLQ except the instance with parameter $k = 50$. For another 84 out of 102 larger graphs, TOPKWCLQ can also obtain good objective values where CPLEX failed.

Based on the benchmark introduced in Section 4.1, Table 7 summarizes the computational results of CPLEX and TOPKWCLQ on 102 real-world graphs. From Table 7, we observe that for almost all instances under the five values of the parameter k, our TOPKWCLQ algorithm can obtain better solutions than the lower bound of CPLEX. It indicates the superiority of the proposed algorithm TOPWCLQ.

Table 7. Summary of comparison between CPLEX and TOPKWCLQ on real-world graphs. #Better denotes the number of graphs where an algorithm finds better objective values. #N/A denotes the number of graphs where an algorithm fails to find an objective value.

Benchmark	k	CPLEX		TOPKWCLQ	
		#Better	#N/A	#Better	#N/A
real-world graphs (102)	10	5	84	87	0
	20	2	85	97	0
	30	1	86	97	0
	40	1	86	97	0
	50	0	87	98	0

5. Conclusions

In this paper, we propose the diversified top-k weight clique search problem and formalize DTKC and DTKWC search problem. The scoring strategy is proposed to find diversified maximal weight cliques for our algorithm. A local search algorithm for the DTKWC search problem based on the scoring strategy and random restart strategy is then proposed, called TOPKWCLQ. This algorithm interleaves maximal weight clique set construction and updating. Experiments on the real-world benchmark show the effectiveness and efficiency of our algorithm. Moreover, further work is to investigate the enhanced configuration checking strategy used in [2] to enhance the performance of the algorithm.

Author Contributions: Methodology, J.W.; software, J.W.; writing—original draft preparation, J.W.; writing—review and editing, J.W. and M.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Fundamental Research Funds for the Central Universities 2412018QD022, NSFC (under Grant No. 61976050, 61972384) and Jilin Provincial Science and Technology Department under Grant No. 20190302109GX.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhou, J.; Li, C.; Zhou, Y.; Li, M.; Liang, L.; Wang, J. Solving diversified top-k weight clique search problem. *Sci. China Inf. Sci.* **2021**, *64*, 150105. [CrossRef]
2. Wu, J.; Li, C.; Jiang, L.; Zhou, J.; Yin, M. Local search for diversified Top-*K* Clique Search Probl. *Comput. Oper. Res.* **2020**, *116*, 104867. [CrossRef]

3. Zhou, Z.; Xiao, Z.; Deng, W. Improved community structure discovery algorithm based on combined clique percolation method and K-means algorithm. *Peer-to-Peer Netw. Appl.* **2020**, *13*, 2224–2233. [CrossRef]
4. Pelofske, E.; Hahn, G.; Djidjev, H. Solving large maximum clique problems on a quantum annealer. In *International Workshop on Quantum Technology and Optimization Problems*; Springer: Berlin/Heidelberg, Germany, 2019, pp. 123–135.
5. Chang, L. Efficient maximum clique computation and enumeration over large sparse graphs. *VLDB J.* **2020**, *29*, 999–1022. [CrossRef]
6. Jiang, H.; Li, C.; Manyà, F. An Exact Algorithm for the Maximum Weight Clique Problem in Large Graphs. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Singh, S.P., Markovitch, S., Eds.; AAAI Press: Menlo Park, CA, USA, 2017; pp. 830–838.
7. Yuan, L.; Qin, L.; Lin, X.; Chang, L.; Zhang, W. Diversified top-k clique search. In Proceedings of the 31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, Korea, 13–17 April 2015; Gehrke, J., Lehner, W., Shim, K., Cha, S.K., Lohman, G.M., Eds.; IEEE Computer Society: Washington, DC, USA, 2015; pp. 387–398. [CrossRef]
8. Yuan, L.; Qin, L.; Lin, X.; Chang, L.; Zhang, W. Diversified top-k clique search. *VLDB J.* **2016**, *25*, 171–196. [CrossRef]
9. Lee, C.; Reid, F.; McDaid, A.; Hurley, N. Detecting highly overlapping community structure by greedy clique expansion. *arXiv* **2010**, arXiv:1002.1827.
10. Zheng, X.; Liu, T.; Yang, Z.; Wang, J. Large cliques in Arabidopsis gene coexpression network and motif discovery. *J. Plant Physiol.* **2011**, *168*, 611–618. [CrossRef] [PubMed]
11. Jiang, H.; Li, C.; Liu, Y.; Manyà, F. A Two-Stage MaxSAT Reasoning Approach for the Maximum Weight Clique Problem. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, LA, USA, 2–7 February 2018; McIlraith, S.A., Weinberger, K.Q., Eds.; AAAI Press: Palo Alto, CA, USA, 2018; pp. 1338–1346.
12. Li, C.M.; Liu, Y.; Jiang, H.; Manyà, F.; Li, Y. A new upper bound for the maximum weight clique problem. *Eur. J. Oper. Res.* **2018**, *270*, 66–77. [CrossRef]
13. Jain, S.; Seshadhri, C. The power of pivoting for exact clique counting. In Proceedings of the 13th International Conference on Web Search and Data Mining, Houston, TX, USA, 3–7 February 2020; pp. 268–276.
14. Wang, Y.; Cai, S.; Chen, J.; Yin, M. SCCWalk: An efficient local search algorithm and its improvements for maximum weight clique problem. *Artif. Intell.* **2020**, *280*, 103230. [CrossRef]
15. Sevinc, E.; Dokeroglu, T. A novel parallel local search algorithm for the maximum vertex weight clique problem in large graphs. *Soft Comput.* **2020**, *24*, 3551–3567. [CrossRef]
16. Wu, J.; Yin, M. Local Search for Diversified Top-k s-plex Search Problem (Student Abstract). In Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, 2–9 February 2021; AAAI Press: New Orleans, LA, USA, 2021; pp. 15929–15930.
17. Cai, S. Balance between Complexity and Quality: Local Search for Minimum Vertex Cover in Massive Graphs. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, 25–31 July 2015; Yang, Q., Wooldridge, M.J., Eds.; AAAI Press: Palo Alto, CA, USA, 2015; pp. 747–753.
18. Rossi, R.A.; Ahmed, N.K. The Network Data Repository with Interactive Graph Analytics and Visualization. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; Bonet, B., Koenig, S., Eds.; AAAI Press: Palo Alto, CA, USA, 2015; pp. 4292–4293.
19. Cai, S.; Lin, J. Fast Solving Maximum Weight Clique Problem in Massive Graphs. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9–15 July 2016; Kambhampati, S., Ed.; IJCAI/AAAI Press: Palo Alto, CA, USA, 2016; pp. 568–574.
20. Fan, Y.; Li, N.; Li, C.; Ma, Z.; Latecki, L.J.; Su, K. Restart and Random Walk in Local Search for Maximum Vertex Weight Cliques with Evaluations in Clustering Aggregation. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, 19–25 August 2017; Sierra, C., Ed.; AAAI Press: Palo Alto, CA, USA, 2017; pp. 622–630. [CrossRef]
21. Nogueira, B.C.S.; Pinheiro, R.G.S. A CPU-GPU local search heuristic for the maximum weight clique problem on massive graphs. *Comput. Oper. Res.* **2018**, *90*, 232–248. [CrossRef]
22. Lin, J.; Cai, S.; Luo, C.; Su, K. A Reduction based Method for Coloring Very Large Graphs. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, 19–25 August 2017; Sierra, C., Ed.; AAAI Press: Palo Alto, CA, USA, 2017; pp. 517–523. [CrossRef]
23. Gao, J.; Chen, J.; Yin, M.; Chen, R.; Wang, Y. An Exact Algorithm for Maximum k-Plexes in Massive Graphs. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, 13–19 July 2018; Lang, J., Ed.; AAAI Press: Palo Alto, CA, USA, 2018; pp. 1449–1455. [CrossRef]