HEA-D: A Hybrid Evolutionary Algorithm for Diversified Top-k Weight Clique Search Problem

Jun Wu[1] , **Chu-Min Li**[2] , **Yupeng Zhou**[1] , **Minghao Yin**[1,3*] , **Xin Xu**[1] and **Dangdang Niu**[4]
[1]Information Science and Technology, Northeast Normal University, China
[2]MIS, Universitéde Picardie Jules Verne, France
[3]Key Laboratory of Applied Statistics of MOE, Northeast Normal University, China
[4]College of Information Engineering, Northwest A&F University, China
{wuj342, ymh, zhouyp605, xux894}@nenu.edu.cn, chu-min.li@u-picardie.fr, niudd@nwafu.edu.cn

Abstract

The diversified top-k weight clique (DTKWC) search problem is an important generalization of the diversified top-k clique (DTKC) search problem with extensive applications, which extends the DTKC search problem by taking into account the weight of vertices. In this paper, we formulate the DTKWC search problem using mixed integer linear program constraints and propose an efficient hybrid evolutionary algorithm (HEA-D) which combines a clique-based crossover operator and an effective simulated annealing-based local optimization procedure to find high-quality local optima. The experimental results show that HEA-D performs much better than the existing methods on two representative real-world benchmarks.

1 Introduction

A clique is a subset of vertices from an unweighted graph, where any two distinct vertices in the subset are adjacent and if it is not covered by any other clique of the given graph, it is a maximal clique. The maximum clique problem (MCP) is to find a maximal clique with the largest cardinality in the graph. Moreover, the maximum weight clique problem (MWCP) is a generalization of MCP with a positive integer assigned to each vertex as its weight value. MWCP aims to find a maximal weight clique which is not a subset of any other maximal weight clique and the total weight of the clique is maximum. MCP and MWCP are NP-hard problems [Garey and Johnson, 1979] with a wide range of practical applications in different fields, including combinatorial auction [Zhou *et al.*, 2020], community detection [Chang, 2020], and video object segmentation [Jiang *et al.*, 2017].

Note that the MCP and MWCP only take into account the maximum one. In order to have larger and more informative solutions, the diversified top-k clique (DTKC) search problem is introduced to find k maximal cliques to cover as many vertices as possible which are not only individually large but also lowly overlapping with each other, and k is an integer parameter which requires to be provided. This problem is NP-hard [Wu *et al.*, 2020]. DTKC search problem is notable for its numerous applications, such as the community search in social networks [Lee *et al.*, 2010], motif discovery in molecular biology [Zheng *et al.*, 2011], and anomaly detection in complex networks [Berry *et al.*, 2004]. The concept DTKC was first introduced by [Yuan *et al.*, 2015].

However, in many situations, the vertices of a graph do not have the same weight, so the DTKC model is not sufficient. So we focus on the DTKWC search problem, which attempts to find k maximal weight cliques of a graph, in the sense that they cover the most weight in the graph. Since the DTKC is just a particular case of DTKWC in which the weight of all vertices is the same, the DTKWC is also NP-hard. Unlike MCP and MWCP, which have been investigated by many researchers, only a few algorithms have been proposed for solving the DTKC and DTKWC search problems.

Several approximation algorithms based on clique enumeration methods were proposed to solve the DTKC search problem [Yuan *et al.*, 2015], and [Wu *et al.*, 2020] provide a local search algorithm (TOPKLS) with two novel heuristic strategies allowing to obtain high-quality solutions of the DTKC search problem. For the DTKWC search problem studied in this work, the exact and heuristic algorithms have been proposed by [Zhou *et al.*, 2021], transforming the DTKWC search problem to the weighted partial MaxSAT (WPMS) problem with direct encoding (DE) and independent set partition based encoding (ISPE), which are then solved by the state-of-the-art WPMS solvers. However, due to the NP-hardness of the DTKWC search problem, the above encoding methods failed to encode large or even medium-sized graphs to WPMS. Motivated by this, [Wu and Yin, 2021] proposed a heuristic algorithm (TOPKWCLQ) to solve the DTKWC search problem on large graphs, which repeats a local search procedure within a reasonable time.

We note that the existing approaches have difficulties to robustly and consistently produce high-quality solutions for large instances. Furthermore, to the best of our knowledge, the powerful population-based evolutionary approach for the DTKWC search problem has not been investigated. Therefore, in this work, we fill this gap by presenting an effective hybrid evolutionary algorithm called HEA-D (hybrid evolutionary algorithm for DTKWC). The main contributions of this work are summarized as follows.

First, from the algorithm perspective, this is the first time a hybrid evolutionary algorithm has been used to solve the

*Corresponding author

DTKWC search problem with several powerful components. The clique-based crossover operator is carefully designed to generate better offspring solutions during the search period. And this operator requires a score function to evaluate the quality of the cliques in the current solution. Furthermore, the simulated annealing-based local optimization is used to find local optimal solutions by effectively exploring the candidate solutions which are the restricted neighborhood of the current solution. Meanwhile, the pool of the population always maintains different and feasible solutions. Second, from the computational perspective, the experimental results demonstrate that our algorithm obtains the best solutions on the 550 benchmark instances tested and is often more than 1 to 3 orders of magnitude faster than the competitors on many instances. In particular, the new record results for 334 out of 550 benchmark instances (for 334 large real-world instances, while for small instances, all algorithms find the optimal solutions) are reported. Finally, HEA-D starts its search with an initial population obtained with a greedy construction procedure, and this population size is no more than a given constant value. Therefore, this work shows the benefit of the population-based approach for solving the DTKWC search problem, and this method can clearly also be used to solve other diversified top-k cohesive subgraph problems.

2 Diversified Top-k Weight Clique Search Problem

2.1 Preliminaries

A weighted graph $G = (V, E, w)$ is a graph with $|V|$ vertices, $|E|$ edges, and a weight function w that assigns to each vertex v_i of V a non-negative integer $w(v_i)$ representing its weight. A weight clique c in G is a set of vertices such that for any $u, v \in c$, $(u \neq v)$, 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 is a *maximal weight clique* if and only if there exists no clique $c^{'}$ in G such that $c \subset c^{'}$ and $\omega(c) < \omega(c^{'})$. Given a set of maximal weighted cliques $C = \{c_1, c_2, \dots\}$, the *coverage* of a weight clique set C is defined as $cov(C) = \bigcup_{c_i \in C} c_i$, while the *weight* of C is the total weight of the vertices in G covered by the maximal weight cliques in C, denoted by $W(C)$ and calculated by the formula $W(C) = \sum_{v \in (\bigcup_{c \in C} c)} w(v)$. The *private vertices* of a maximal weight 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., $priv(c, C) = c \setminus cov(C \setminus c)$.

Definition 1 (Diversified top-k weight clique, DTKWC). *Given a weighted graph $G(V, E, w)$ and a fixed integer k, the diversified top-k weight clique search problem is to compute a set C, so 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.*

Let $G(V, E, w)$ be a given graph and C is the current solution. Suppose that $S = \{C_1, C_2, \dots\}$ is the search space including all feasible solutions of the given problem, i.e., the DTKWC search problem in this paper. Thus, each item in S contains at most k maximal weight cliques of G.

Definition 2 (Distance). *The distance between C_i and C_j, denoted by $Dist(C_i, C_j)$, is the minimum number of vertices that need to be changed in C_i such that the resulting solution becomes the same as C_j. If $Dist(C_i, C_j) > 0$, then C_i and C_j are not identical solutions. Otherwise, they have the same coverage.*

This distance $Dist(C_i, C_j)$ can be calculated in polynomial time [Porumbel *et al.*, 2011].

2.2 Constraint Model for DTKWC Search Problem

Given a graph $G = (V, E, w)$, the DTKWC search problem involves finding at most k maximal weight cliques that cover maximum weight of vertices in a given graph. Let x_{ih} be the binary variable such that $x_{ih} = 1$ if the vertex v_i is allocated to the h^{th} maximal weight clique, and $x_{ih} = 0$ otherwise. Likewise, let y_i be a binary variable associated with vertex v_i such that $y_i = 1$ if v_i in a maximal weight clique, $y_i = 0$ otherwise. w_i denotes the weight of the vertex v_i. $\bar{E}$ is a set of edges in the complement graph $\bar{G} = (V, \bar{E}, w)$. The DTKWC search problem can be expressed by the following MILP constraint formulation.

$$\max f(G) = \sum_{i=1}^{|V|} y_i w_i \tag{1}$$

$$\text{subject to: } x_{ih} + x_{jh} \leq 1 \quad \forall (v_i, v_j) \in \bar{E}, h \in [1, k] \tag{2}$$

$$y_i = \begin{cases} 0, & \text{if } \sum_{h=1}^{k} x_{ih} = 0, \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

$$x_{ih} \in \{0, 1\} \quad \forall i \in [1, |V|], h \in [1, k] \tag{4}$$

In the above formulation, the objective function (1) is to maximize the function $f(G)$, i.e. maximize the total weight of the selected maximal weight cliques. The intuitions of other formulas are given below. 1) Constraint (2) guarantees that there is an edge between every two vertices in a clique; 2) Constraint (3) means that the value of $y_i = 1$ if there exists at least one $x_{ih} = 1$ ($v_i \in V$ and $1 \leq h \leq k$); otherwise, the value of $y_i = 0$; 3) Constraint (4) indicates that x_{ih} ($v_i \in V$ and $1 \leq h \leq k$) can only be assigned a value of 0 or 1.

3 HEA-D: the Top-Level Algorithm

This section describes a hybrid evolutionary algorithm, called HEA-D, for the DTKWC search problem on top level. Details of important components will be presented in the following.

As shown in Algorithm 1, HEA-D starts with an initial population Pop of at most p (p is a parameter and $p \geq 2$) individuals or solutions (line 1). After that, HEA-D performs a number of generations to evolve the population until the given stopping condition is met (lines 3-8). In this work, the condition is a cutoff time limit. During each generation, HEA-D picks out two parent solutions C_1 and C_2 from Pop. A clique-based crossover operator (Crossover) is used to recombine C_1 and C_2 to generate an offspring solution (line 5). Generally, the quality of this offspring solution is not good enough. Thus, a simulated annealing-based local optimization (SALO) procedure is performed to improve its quality (line 6). Finally, the improved offspring solution is used to update the population according to the updating rules (line 8).

Algorithm 1: The main framework of the HEA-D

Input: a weighted graph $G(V, E, w)$, one integer k, population size p and other parameters of the algorithm
Output: a set C^* containing at most k maximal weight cliques

1 $Pop \leftarrow$ PopInitialize(p);
2 $C^* \leftarrow \arg\max_{C \in Pop} W(C)$;
3 **while** *stopping condition is not met* **do**
4 Randomly select two solutions C_1, C_2 from Pop;
5 $C_0 \leftarrow$ Crossover(C_1, C_2); /* Section 4 */
6 $C_0 \leftarrow$ SALO(C_0); /* Section 5 */
7 **if** $W(C_0) > W(C^*)$ **then** $C^* \leftarrow C_0$;
8 $Pop \leftarrow$ UpdatePool(Pop, C_0);
9 **return** C^*;

Below, we describe the procedures of population initialization and updating.

3.1 Population Initialization

The initial population Pop consists of no more than p feasible solutions, and each solution includes at most k maximal weight cliques. Pop is built using the population initialization procedure. Let C_0 denote a candidate solution initialized as an empty set. The initialization procedure creates at most k maximal weight cliques. If one clique is not empty, add it into C_0 directly; otherwise, the inner loop is ended. After the trivial solution C_0 is created, the SALO procedure is invoked to improve C_0, and then add C_0 into Pop. Note that the constructing and optimization procedures of a new solution are repeated p times to fill Pop with p improved feasible solutions. However, all improved solutions in Pop may be different from each other, or some of them are identical solutions. In addition, it is time-consuming and unnecessary to have repeated elements in the population. Consequently, we use the distance measure (Definition 2) to calculate the pairwise distances of the p solutions and eliminate duplicates. If all solutions in Pop are identical, i.e. $|Pop| = 1$, we try to create another different solution in the sense of the distance and apply again the SALO procedure to improve it. Therefore, this initialization procedure can always create an initial population with 2 to p high-quality and diverse solutions.

3.2 Population Updating

For keeping both high-quality and diversity of the population Pop, the population need to be updated when an offspring individual that is a feasible solution for the DTKWC search problem is generated and further improved by Crossover and SALO procedures, respectively. We first calculate the distance between the new offspring individual C_0 and each one in Pop. Thus, there are three situations in which the population is tried to update with the new offspring individual: (1) If C_0 is identical to an individual in the current population Pop, C_0 is dropped without changing the population. (2) If C_0 is different from all individuals in the population and the size of the current population is less than p, C_0 will be included directly into the population. (3) If neither of the above situations is satisfied. We compare C_0 with the worst individual C_{min} that has the least weight in the current population. If the new offspring C_0 is better than the worst one C_{min}, replace C_{min} with this offspring; otherwise, drop C_0 directly.

Additionally, as the current best solution is not improved in L steps, $|Pop| \times \theta_{reduce}$ (θ_{reduce} is a parameter of HEA-D) individuals in Pop will be removed, allowing more opportunities for the diversity of individuals to be added to Pop.

Algorithm 2: Crossover

Input: two selected parent solutions C_1 and C_2
Output: a offspring solution C_0

1 $C_0 \leftarrow C_1 \cup C_2$;
2 **for** $i = 1$ *to* k **do**
3 **for** $c \in C_0$ **do**
4 Calculate the scoring value $score(c)$ of c according to the score function.
5 $c_{min} \leftarrow \arg\min_{c \in C_0} score(c)$;
6 Remove the maximal weight clique c_{min} which is the one with the least score value from C_0, breaking ties in favour of the new one;
7 **return** C_0;

4 Clique-based Crossover Operator

Generally, the memetic algorithm that integrates the evolutionary algorithm and the local search algorithm needs to design a suitable crossover operator such that this operator allows the offspring solutions to inherit the corresponding genetic information from the parent solutions. For the DTKWC search problem, the maximal weight cliques shared by parents are used to represent such information in HEA-D. Therefore, a well-designed crossover operator based on the cliques is presented.

Specifically, it is important to find a way to evaluate whether the cliques in the parent solutions can be inherited to the offspring solution. Consequently, we propose a score function based on the total weight of the private vertices, following the general principle presented above and described below.

Given a weighted graph $G = (V, E, w)$, a maximal weight clique set C and a maximal weight clique c of G ($c \in C$), the score of c is defined as:

$$score(c) = \sum_{v \in priv(c,C)} w(v) \tag{5}$$

In Algorithm 2, let C_0 denote a new offspring solution and C_1, C_2 be the selected parent solutions. The crossover operator generates a new offspring solution C_0 by inheriting the cliques with better profit from C_1 and C_2 in order to maximize the total weight of the covered vertices included in C_0. The criteria for deciding which clique can be inherited is relying on the score of each clique calculated by the Equation (5). For the above purpose, all cliques in C_1 and C_2 are contained in C_0 which is initialized as an empty set (line 1). Then, the

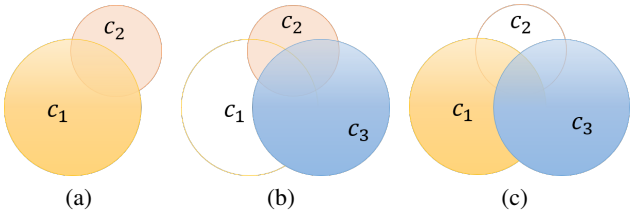

Figure 1: Example of the neighborhood for DTKC search problem. The circles c_1, c_2 and c_3 represent maximal weight cliques in a given graph. The surface of one circle denotes its weight. (a) Current solution, (b) A neighbor solution, (c) Another neighbor solution.

score values for all cliques in C_0 will be calculated according to Equation (5) (line 4). Among them, the clique with the least score value is eliminated from C_0 (lines 5-6). These two sequential steps will be repeated k iteration steps to recombine a new offspring solution from the reference parents, so that C_0 becomes a feasible solution again with the largest weight value. Note that the crossover operator aims to transfer as many cliques with the greatest score value in both reference parents to the new offspring as possible in each iteration.

5 Local Optimization

This section is dedicated to the simulated annealing-based local optimization (SALO), which can be used to improve the solutions created by population initialization and the offspring solution generated by crossover operator. As discussed in the literature, during the search process of memetic algorithms, a local optimization method can find higher-quality solutions than only evolutionary algorithms for constrained optimization problems. Therefore, following this, a SALO procedure is used to effectively explore the candidate solutions. We first introduce the neighborhood used by the SALO and then present the general algorithm.

5.1 Neighborhood

During the SALO procedure, the neighborhood of a feasible solution of the DTKWC search problem is one key ingredient and should be carefully designed. Therefore, in the following, we introduce a new definition of neighborhood for the DTKWC search problem, which is based on the maximal weight cliques. The distance between a feasible solution and its neighbor solution for the DTKWC search problem must be greater than zero, and there is one and only one clique between them that is different. The definition of a neighborhood follows the general principle presented above and is described as follows.

According to the definition of distance between two feasible solutions of the DTKWC search problem, a neighbor solution can be built by exchanging a random maximal weight clique from the current solution with another maximal weight clique. However, with no restriction on the cliques to be exchanged, there are numerous candidate solutions which needs to be checked, and many of them fail to improve the quality of the current solution. To improve the search efficiency, the SALO procedure will accept a *restricted neighborhood* that

Algorithm 3: SALO

Input: a current solution C
Output: a best solution C^{best} found so far

1 $T \leftarrow T_{init}$, $C^{best} \leftarrow C$;
2 **while** *True* **do**
3 **for** $iter = 1$ *to* $p \times k \times \theta_{size}$ **do**
4 $c \leftarrow$ find a new maximal weight clique from the given graph G;
5 $C^{'} \leftarrow C \cup \{c\}$;
6 Compute the score for each clique in $C^{'}$;
7 $c_{min} \leftarrow \arg\min_{c \in C} score(c)$, breaking ties in favor of the new one;
8 $C^{'} \leftarrow C^{'} \setminus c_{min}$;
9 With probability defined in Equation (6), accept solution $C^{'}$ as new current solution C;
10 **if** $W(C) > W(C^{best})$ **then** $C^{best} \leftarrow C$;
11 $T \leftarrow T \times \theta_{cool}$; /* Temperature cooling down */
12 **if** C^{best} *is not updated for* l *rounds* **then** break ;
13 **return** C^{best};

excludes non-promising candidate solutions to replace the old one. For example, in Figure 1, let $C = \{c_1, c_2\}$ be the current solution and c_3 be a new maximal weight clique which is different from the one in C. Suppose that the parameter $k = 2$ and we add c_3 into C, in this case, one of the cliques in C must be removed such that C is also a feasible solution for the DTKWC search problem, which leads to C with the largest coverage of the objective value. Note that $\{c_2, c_3\}$ and $\{c_1, c_3\}$ are both the neighbor solutions of $\{c_1, c_2\}$, and it is easy to check that $\{c_1, c_3\}$ is the best solution among all neighbors of C.

With this restricted neighborhood, the SALO procedure can transfer from the current solution to the best solution possible. It has the advantages of having a smaller size compared with the conventional neighborhood and being more focused on good enough neighbor solutions.

5.2 General Algorithm

The SALO procedure explores the candidate solutions according to the restricted neighborhood above to improve the solutions created by the initialization procedure and crossover operator following the general simulated annealing framework. As shown in Algorithm 3, the SALO procedure performs a number of search rounds (lines 2-12) with different temperature values T (initially set to T_{init}). Let C^{best} be the best solution found by the SALO procedure (initially set to the current solution C). In one round, the SALO procedure checks $p \times k \times \theta_{size}$ candidate restricted neighbor solutions (lines 3-10). Neighbor solutions are generated by adding a new maximal weight clique in C and again removing the worst one, breaking ties in favor of the new one (lines 4-8).

Sometimes, the improved solutions are not always better than the current best solution and a decision must be made to decide whether an unimproved neighbor solution $C^{'}$ is ac-

Parameter	Range	Value
p	{10, 15, 20, 25, 30}	20
θ_{size}	{2, 4, 8, 16, 32}	8
θ_{cool}	{0.90, 0.91, . . . , 0.99}	0.96
θ_{reduce}	{0.1,0.2,. . . ,0.9}	0.5
l	{1,2,. . . ,10}	5
L	{50,100,150,200}	100

Table 1: Parameter setting of HEA-D.

cepted as the new current solution according to the following acceptance probability:

$$Pr\{C \leftarrow C^{'}\} = min\{1, e^{\frac{W(C^{'})-W(C)}{T}}\} \quad (6)$$

If the acceptance probability is satisfied, the current solution C is updated by an unimproved neighbor solution $C^{'}$; otherwise, the search process will continue (line 9). If the current solution can be improved by a neighbor solution, and it is better than C^{best} in this iteration, the best solution C^{best} is updated (line 10). At the end of each search round, as many simulated annealing algorithms, the temperature T will be decreased by a constant factor θ_{cool} which is a parameter of HEA-D (line 11).

Further, the current solution could not be improved from the neighborhood in numerous rounds. To reduce unnecessary search rounds, a new criterion is adopted such that the search process will be terminated and return the best solution found so far when C^{best} is unchanged in l rounds (line 12).

Specifically, the SALO procedure is sensitive to the initial temperature T_{init} which is set using a simple automated binary search. This method considers the frequency of reaching the best solution in one search round. And the initial temperature T_{init} is set to 1000 first (the initial range is [1,2000]). If this frequency is equals to 50%, then set T_{init} to this value; otherwise, according to the value of the frequency is less or greater than 50%, the frequency is set to the first or second half-sized range in the next search round.

6 Computational Experiments

We carry out extensive experiments to evaluate the performance of HEA-D on two representative real-world graphs used in the previous literatures: advertisement putting benchmark (8 graphs) [Zhou *et al.*, 2021] and weighted real-world large graph benchmark (102 graphs) [Wu and Yin, 2021]. We compare HEA-D against the existing algorithms, including a heuristic algorithm TOPKWCLQ [Wu and Yin, 2021] and two exact algorithms DE and ISPE [Zhou *et al.*, 2021].

6.1 Experimental Setup

HEA-D is implemented in C++ and compiled using g++ with "-O3". Detailed results and the sourced code is reported in github[1]. The default values of the parameters used in HEA-D are shown in Table 1 tuned with the automatic configuration tool irace [López-Ibáñez *et al.*, 2016]. All experiments were performed on CentOS with 2.00 GHz CPU and 32 GB memory. The optimal solutions of the advertisement putting instances are known, whereas the weighted large real-world instances have unknown optimal solutions. Moreover, for the advertisement putting problem, the size of the instances is small, so that in some instances all vertices will be covered when k is set to 5 or more for some instances. For this reason, we set the parameter k to 1, 2, 3, 4, and 5, respectively, while for each large real-world graph in our experiments, the parameter k is set to 10, 20, 30, 40, and 50, respectively. Thus, we have $(8 + 102) \times 5 = 550$ DTKWC search problem instances in total. For HEA-D and TOPKWCLQ, they are performed on 10 independent runs with a cutoff time (600 seconds) on each instance. For CPLEX solver and two exact algorithms DE and ISPE, a cutoff time of one hour is used. For advertisement putting instances, the results are not reported here, owing to they are so easy that HEA-D and its competitors find the optimal solutions quickly. For real-world large graphs, we compared HEA-D with the CPLEX solver (version 12.9), which uses the mathematical model presented in Section 2.2 and the heuristic algorithm for the DTKWC search problem called TOPKWCLQ.

[1]https://github.com/wujunzero/HEA-D

6.2 Computational Results

For large real-world instances, HEA-D significantly outperforms the competitors. HEA-D found the best solution on all instances, indicating that HEA-D dominated TOPKWCLQ, CPLEX, DE and ISPE. For 19 out of 510 real-world instances, CPLEX was able to prove the solution optimal, where the values of the lower bound and upper bound are equivalent in the results of CPLEX. For 45 out of 73 instances where CPLEX solver can reach a solution, HEA-D finds better objective values; HEA-D and CPLEX find the same objective values on the remaining 28 instances (Figure 2(a)). Furthermore, considering the average computing time of reaching the best solutions for large real-world instances, we observe that HEA-D requires less time than TOPKWCLQ on most instances and performs 1-3 orders of magnitude faster than TOPKWCLQ for most instances (Figure 2(b)). In addition, the best value and average value found by HEA-D are the same on many instances, indicating that the feasibility and robustness of HEA-D are better than TOPKWCLQ.

To verify whether there exists a statistically significant difference between HEA-D and its competitors in terms of the best (average) objective values, the p-values from the non-parametric Wilcoxon signed ranks test [Derrac *et al.*, 2011; Carrasco *et al.*, 2020] with a significance level of 0.05 are provided in Table 2. Column R^{+}_{best} (R^{+}_{avg}) denotes the sum of ranks for the instances in which HEA-D outperforms the compared algorithms, in terms of the best (average) objective value, while R^{-}_{best} (R^{-}_{avg}) refers to the sum of ranks for the opposite. Therefore, the statistical tests in Table 2 confirm that HEA-D has a significance improvement over the two competitors (CPLEX and TOPKWCLQ) with a significance level of 0.05. Furthermore, the summary results are shown in Table 3. "#Better" and "#Equal" record the number of instances for which the associated algorithm yields a better and equal solution than the best solution produced by the other algorithms, respectively. HEA-D outperformed CPLEX and performed the same as TOPKWCLQ on 175 instances in

Comparison	k	R^+_{best}	R^-_{best}	p-value	R^+_{avg}	R^-_{avg}	p-value
vs. CPLEX	10	89	0	2.59e-16	89	0	2.59e-16
	20	99	0	5.78e-18	99	0	5.78e-18
	30	98	0	8.46e-18	98	0	8.46e-18
	40	98	0	8.46e-18	98	0	8.46e-18
	50	98	0	8.46e-18	98	0	8.46e-18
vs. TOPKWCLQ	10	55	0	1.14e-10	67	1	1.59e-12
	20	63	0	5.29e-12	76	0	3.69e-14
	30	69	0	5.33e-13	82	0	3.73e-15
	40	74	0	7.89e-14	86	1	6.38e-16
	50	74	0	7.89e-14	92	0	8.28e-17

Table 2: Wilcoxon signed ranks test results of HEA-D and the reference algorithms in terms of both the better and the average solutions on 550 instances, with a significance level of 0.05.

Benchmark	k	DE/ISPE	CPLEX		TOPKWCLQ		HEA-D	
			#Better	#Equal	#Better	#Equal	#Better	#Equal
real-world graphs (102)	10	N/A	0	13	0	49	53	49
	20	N/A	0	3	0	39	63	39
	30	N/A	0	4	0	32	69	33
	40	N/A	0	4	0	28	74	28
	50	N/A	0	4	0	28	74	28
Total		0	0	28	0	175	334	176

Table 3: Summary of comparison between CPLEX, TOPKWCLQ and HEA-D on real-world graphs. "N/A" means that the reference algorithm failed to solve the instances.

terms of the best objective values. In particular, HEA-D established new lower bounds for 334 out of the 550 instances.

6.3 Analysis

In this section, we perform additional experiments to analyze two key ingredients of HEA-D: the population-based memetic search framework and the SALO procedure. The experiments were conducted on all large real-world graphs considered in this work and the parameter $k = 10$. To assess the effectiveness of the two key ingredients, we compared HEA-D with two HEA-D variants: HEA-D-Descent and SA-Restart, which replace SALO by a pure descent procedure and keep SALO procedure only, respectively. Consequently, we can verify the importance of SALO and the population-based memetic framework by comparing HEA-D with HEA-D-Descent and SA-Restart, respectively. From the results, we can make the following observations: (1) SA-Restart reports the worst results in terms of best and average objective values, which are significantly worse than the results of HEA-D, indicating that removing the memetic framework (set the population size to 1) drastically degrades the performance of the HEA-D and the memetic framework is one key component that ensuring HEA-D's high performance; (2) HEA-D-Descent reports better results than HDA-D-Restart, but its results are still significantly worse than the results of HEA-D, indicating that disabling the SALO procedure negatively impacts the performance of HEA-D, i.e., SALO procedure positively contributes to the high performance of the HEA-D.

In addition, the Wilcoxon signed ranks test results from

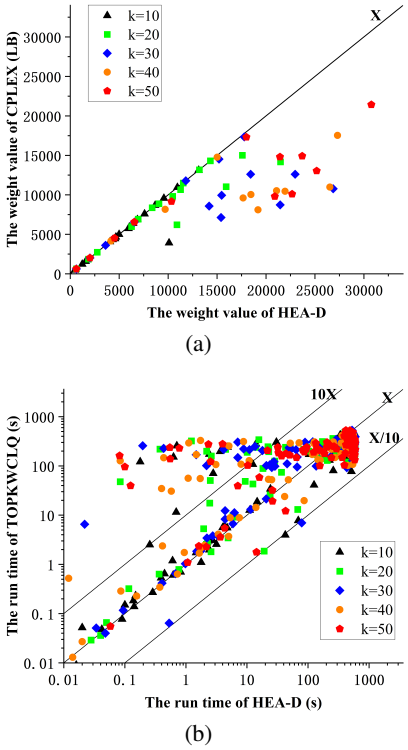

Figure 2: The comparisons of HEA-D and its competitors. (a) The comparisons of weight values on the instances which CPLEX can solve; (b) The comparisons of run time on all tested instances.

Comparison	R^+_{best}	R^-_{best}	p-value	R^+_{avg}	R^-_{avg}	p-value
vs. HEA-D-Descent	54	0	1.67e-10	64	0	3.61e-12
vs. SA-Restart	81	0	5.46e-15	94	1	6.42e-17

Table 4: Wilcoxon signed ranks test results of HEA-D and its two variants HEA-D-Descent and SA-Restart on 110 instances, with a significance level of 0.05.

Table 4 indicate that HEA-D performs better than HEA-D-Descent and SA-Restart with a significance level of 0.05.

7 Conclusions

This paper proposed a hybrid evolutionary algorithm for the DTKWC search problem that is proved to be an NP-hard problem, called HEA-D. On the basis of the general memetic framework, HEA-D combines a clique-based crossover operator for solution recombination (offspring generation) and a SALO procedure for effective local optimization. Computational results on a wide range of real-world instances demonstrated the superiority of HEA-D over the reference methods. This work takes a first step towards the hybrid evolutionary algorithm for the DTKWC search problem and provides new insights into the memetic framework for diversified top-k cohesive subgraph problems as well as related clique problems.

Acknowledgements

This work is supported by the Fundamental Research Funds for the Central Universities 2412019ZD013, NSFC (under Grant No.61976050, 61972384). The authors also thank the Matrics Platform of the Université de Picardie Jules Verne.

References

[Berry *et al.*, 2004] Nina Berry, Teresa Ko, Tim Moy, Julienne Smrcka, Jessica Turnley, and Ben Wu. Emergent clique formation in terrorist recruitment. In *The AAAI-04 Workshop on Agent Organizations: Theory and Practice*, pages 1198–1208, 2004.

[Carrasco *et al.*, 2020] Jacinto Carrasco, Salvador García, María del Mar Rueda, S. Das, and Francisco Herrera. Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review. *Swarm Evol. Comput.*, 54:100665, 2020.

[Chang, 2020] Lijun Chang. Efficient maximum clique computation and enumeration over large sparse graphs. *VLDB J.*, 29(5):999–1022, 2020.

[Derrac *et al.*, 2011] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.*, 1(1):3–18, 2011.

[Garey and Johnson, 1979] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[Jiang *et al.*, 2017] Hua Jiang, Chu-Min Li, and Felip Manyà. An exact algorithm for the maximum weight clique problem in large graphs. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 830–838. AAAI Press, 2017.

[Lee *et al.*, 2010] Conrad Lee, Fergal Reid, Aaron McDaid, and Neil Hurley. Detecting highly overlapping community structure by greedy clique expansion. *arXiv preprint arXiv:1002.1827*, 2010.

[López-Ibáñez *et al.*, 2016] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[Porumbel *et al.*, 2011] Daniel Cosmin Porumbel, Jin Kao Hao, and Pascale Kuntz. An efficient algorithm for computing the distance between close partitions. *Discrete Applied Mathematics*, 159(1):53–59, 2011.

[Wu and Yin, 2021] Jun Wu and Minghao Yin. A restart local search for solving diversified top-k weight clique search problem. *Mathematics*, 9(21), 2021.

[Wu *et al.*, 2020] Jun Wu, Chu-Min Li, Lu Jiang, Junping Zhou, and Minghao Yin. Local search for diversified top-*k* clique search problem. *Comput. Oper. Res.*, 116:104867, 2020.

[Yuan *et al.*, 2015] Long Yuan, Lu Qin, Xuemin Lin, Lijun Chang, and Wenjie Zhang. Diversified top-k clique search. In Johannes Gehrke, Wolfgang Lehner, Kyuseok Shim, Sang Kyun Cha, and Guy M. Lohman, editors, *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 387–398. IEEE Computer Society, 2015.

[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.

[Zhou *et al.*, 2020] Zhou Zhou, Zhuopeng Xiao, and Weihong Deng. Improved community structure discovery algorithm based on combined clique percolation method and k-means algorithm. *Peer Peer Netw. Appl.*, 13(6):2224–2233, 2020.

[Zhou *et al.*, 2021] Junping Zhou, Chumin Li, Yupeng Zhou, Mingyang Li, Lili Liang, and Jianan Wang. Solving diversified top-k weight clique search problem. *Sci. China Inf. Sci.*, 64(5), 2021.