Local Search for Diversified Top-k s-plex Search Problem (Student Abstract)

Jun Wu, Minghao Yin*
School of Information Science and Technology, Northeast Normal University,
Changchun, 130117, China
{wuj342,ymh}@nenu.edu.cn

Abstract

The diversified top-k s-plex (DTKSP) search problem aims to find k maximal s-plexes that cover the maximum number of vertices with lower overlapping in a given graph. In this paper, we first formalize the diversified top-k s-plex search problem and prove the NP-hardness of it. Second, we proposed a local search algorithm for solving the diversified top-k s-plex search problem based on some novel ideas. Experiments on real-world massive graphs show the effectiveness of our algorithm.

Introduction

Recently, many researchers have studied some versions of the diversified top-k search problem because of its wide applications in which researchers always have a need for obtaining diverse high-quality solutions to increase flexible results. Among these different versions, the diversified top-k clique (DTKC) search problem (Wu et al. 2020) has already been applied in different fields. However, lots of real-world applications in network analysis and data mining fail to model the clique, due to the strong conditions of the clique itself. Thus, in this paper, we mainly focus on solving the diversified top-k s-plex (DTKSP) search problem, which can be seen as a generalization version of the DTKC search problem. For example, soc-twitter-follows (Rossi and Ahmed 2015) is a social media follow relationship that includes a follow network and a dataset that contains a list of all user-to-user links. We set users as vertices and user-to-user links as edges in a graph G. Therefore, a maximal s-plex in G represents a core group and some important followers; the DTKSP in the given graph represents the top-k maximal divisive core groups and their followers. This can facilitate determining the top-k core groups with great influence on Twitter.

In this paper, we first construct the formulation for the DTKSP search problem and then prove its NP-hardness. Moreover, we design an encoding for this problem and then propose an effective local search algorithm. The experiments are carried out on a wide range of massive graphs to evaluate the effectiveness of our algorithm.

*Corresponding Author

Diversified Top-k s-plex Search Problem

Given an undirected graph $G = (V, E)$ with a vertices set V and a edges set E, where $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. Let $N(u)$ denote a set of neighbours of the vertex u, i.e., $N(u) = \{v|(v, u) \in E \; and \; v, u \in V\}$, and $N[u] = N(u) \cup \{u\}$. Given a graph G, s-plex is a subset $D \subseteq V$ such that each vertex in D must be nonadjacent to at most s vertices in the subgraph induced by D. A maximal s-plex is an s-plex that cannot be extended to any other s-plex of G. For a set of maximal s-plexes $S = \{c_1, c_2, \ldots\}$, the *coverage* of S, denoted by $cov(S)$, is the set of vertices covered by S, i.e., $cov(S) = \bigcup_{c_i \in S} c_i$. The *private vertices* of a maximal s-plex c, denoted by $priv(c, S)$, is a subset of vertices of c not contained in any other s-plex in S, i.e., $priv(c, S) = c \setminus cov(S \setminus c)$. Given two integers k and s, the DTKSP search problem is to find a set of S of at most k maximal s-splexes with the largest coverage $cov(S)$ and keep a lower *overlapping* $cov(S) \setminus \cup_{c_i \in S} priv(c_i, S)$.

Constraint Formulation for the DTKSP Search Problem

The DTKSP search problem involves finding at most k s-plexes of maximum cardinality with lower overlapping from a given graph. It can be formulated as a mixed integer linear program (MILP) as follows:

$$OBJ1 : \max \quad \omega_1(G) = \sum_{i \in V} X_i \tag{1}$$

$$OBJ2 : \min \; \omega_2(G) = \frac{\sum_{h=1}^{k} \sum_{i \in V} x_{ih} - \sum_{i \in V} X_i}{k - 1} \tag{2}$$

subject to:

$$\sum_{j \in V \setminus N[i]} x_{jh} \leq (s-1)x_{ih} + \bar{d}_i(1 - x_{ih}), \quad \forall i \in V, 1 \leq h \leq k \tag{3}$$

$$X_i \leq \sum_{i \in V} x_{ih}, \quad \forall i \in V, 1 \leq h \leq k \tag{4}$$

$$x_{ih} \in \{0, 1\}, \quad \forall i \in V, 1 \leq h \leq k \tag{5}$$

$$X_i \in \{0, 1\}, \quad \forall i \in V \tag{6}$$

where x_{ih} is the binary variable associated with vertex i, such that $x_{ih} = 1$ if vertex i is in the h'th s-plex, $x_{ih} = 0$ otherwise. $\bar{d}_i = |V \setminus N[i]|$ denotes the degree of vertex i in the complement graph $\bar{G} = (V, \bar{E})$. X_i is also a binary variable associated with vertex i. $X_i = 1$ if there exists a vertex i in an s-plex, $X_i = 0$ otherwise.

According to the definition of the DTKSP search problem, this problem needs two objective functions. The first (*OBJ1*) is to keep at most k s-plexes with maximum cardinality. And, the second (*OBJ2*) is to find a lower overlapping solution under the premise of objective *OBJ1*.

NP-hardness of the DTKSP Search Problem

Proposition 1 *The DTKSP search problem is an NP-hard problem.*

Proof. We prove Proposition 1 by reducing one NP-hard problem to the DTKSP search problem. As is well known, the max k-cover problem is a classical NP-hard problem. Given two integers k and s, a set U of n elements, and a collection $S = \{S_1, S_2, \ldots, S_h\}$ of h subsets of U, the max k-cover problem is to select k subsets from S such that their union has the maximum cardinality. We can reduce the max k-cover problem to the DTKSP search problem as follows.

1. For each element in U, create a vertex.
2. Create at least $(|S_i| - s)$ edges connecting with others for each vertex in each subset S_i.

Clearly, the above procedure is polynomial. In this case, solving the DTKSP search problem is equivalent to finding k subsets of S such that their union has the maximum cardinality and then keeping the crossover between the k sets is minimum. Thus, we can conclude that the DTKSP search problem is an NP-hard problem. □

The TOPKSPLEX Algorithm

As shown in Algorithm 1, it consists of the construction (line 3) and updating stages (lines 5-12). Each time, the algorithm constructs a new s-plex starting from the candidate vertices $(V \setminus S)$, S is the current solution. Then TOPKSPLEX tries to add this new s-plex into the current solution S and remove the worst one by calculating the value of $|priv(c_i, S)|$ as a score for each s-plex in S (lines 7-9). If the current solution S can not be updated in L steps, the inner loop will be terminated. After that, if a better solution is found, S^* is updated by S (line 12). Then restart the algorithm until the cutoff time is out. In our experiments, L is initialized to 1000.

Algorithm 1: TOPKSPLEX($G, s, k, cutoff$)

Input: Problem instance $G(k, s)$, the maximum allowed iteration in the update search L, $cutoff$ time
Output: a set S^* containing at most k maximal s-plexes with a lower overlapping

1 $S^* \leftarrow \emptyset$;
2 **while** *elapsed time* $< cutoff$ **do**
3 Constructing a good quality solution S with at most k s-plexes;
4 $S' \leftarrow S, l \leftarrow 0$;
5 **while** $l < L$ **do**
6 Constructing a new s-plex c_{new};
7 $S' = S' \cup \{c_{new}\}$;
8 Updating the score for each s-plex in S';
9 $S' \leftarrow S' \setminus c_{worst}$;
10 **if** $|cov(S')| > |cov(S)|$ **then** $S \leftarrow S', l \leftarrow 0$;
11 **else** $l \leftarrow l + 1$;
12 **if** $|cov(S)| > |cov(S^*)|$ *or* $(|cov(S)| = |cov(S^*)|$ *and* $|\cup_{c_i \in S} priv(c_i, S)| > |\cup_{c_j \in S^*} priv(c_j, S^*)|)$ **then** $S^* \leftarrow S$;
13 **return** S^*; /* return the best solution found */

Experimental Evaluation

We carry out experiments in TOPKSPLEX on real-world massive graphs (Rossi and Ahmed 2015). We downloaded the benchmarks from the author's website[1]; these have recently been used in testing the performance of local search methods. Due to there is no other algorithm that can be compared, we used the solutions obtained from the CPLEX solver (version 12.9) as a reference on the solution quality with the mathematical model presented in this paper. For each instance, our algorithm is performed on 10 independent runs with a cutoff time (360s). For CPLEX solver, a cutoff time of one hour was used.

[1]http://lcs.ios.ac.cn/~caisw/graphs.html

Benchmark		TOPKSPLEX		CPLEX	
		#Better	#N/A	#Better	#N/A
Massive(139)	s=2	537	0	17	51
	s=3	536	0	6	51
	s=4	538	0	3	51
	s=5	534	0	1	51

Table 1: Summary of comparison between TOPKSPLEX and CPLEX on massive graphs.

There are 139 graphs in our experiments. We set the parameter k to 10, 20, 30, 40 and parameter s to 2, 3, 4, 5 for each graph. Hence, we totally have 2224 DTKSP search problem instances. The summary results are shown in Table 1. #Better and #N/A indicate the number of instances where an algorithm finds better solutions or fails to solve the instances, respectively. CPLEX can solve 2020 instances. TOPKSPLEX obtains better or same solutions than solutions as CPLEX on 1993 instances. On the other 27 instances, TOPKSPLEX obtains the solutions with small gaps.

References

Rossi, R.; and Ahmed, N. 2015. The network data repository with interactive graph analytics and visualization. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Wu, J.; Li, C.-M.; Jiang, L.; Zhou, J.; and Yin, M. 2020. Local search for diversified Top-k clique search problem. *Computers & Operations Research* 116: 104867.