Computers and Operations Research 86 (2017) 65–78

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Frequency-driven tabu search for the maximum *s*-plex problem

Yi Zhou[a], Jin-Kao Hao[a,b,*]

[a] *LERIA, University of Angers, 2 bd Lavoisier, 49045 Angers, France*
[b] *Institut Universitaire de France, 1 rue Descartes, 75231 Paris, France*

ARTICLE INFO

Article history:
Received 8 September 2015
Revised 3 May 2017
Accepted 3 May 2017
Available online 6 May 2017

Keywords:
Clique relaxation
Heuristic
Massive network
s-plex

ABSTRACT

The maximum *s*-plex problem is an important model for social network analysis and other studies. In this study, we present an effective frequency-driven multi-neighborhood tabu search algorithm (FD-TS) to solve the problem on very large networks. The proposed FD-TS algorithm relies on two transformation operators (*Add* and *Swap*) to locate high-quality solutions, and a frequency-driven perturbation operator (*Press*) to escape and search beyond the identified local optimum traps. We report computational results for 47 massive real-life (sparse) graphs from the SNAP Collection and the 10th DIMACS Challenge, as well as 52 (dense) graphs from the 2nd DIMACS Challenge (results for 48 more graphs are also provided in the Appendix). We demonstrate the effectiveness of our approach by presenting comparisons with the current best-performing algorithms.

1. Introduction

Given a simple undirected graph $G=(V,E)$ with a set of vertices V and a set of edges E, let $N(v)$ denote the set of vertices adjacent to v in G. Then, an *s*-plex for a given integer $s \geq 1$ ($s \in \mathbb{Z}^+$) is a subset of vertices $C \subseteq V$ that satisfies the following condition: $\forall v \in C$, $|N(v) \cap C| \geq |C| - s$. Thus, each vertex of an *s*-plex C must be adjacent to at least $|C| - s$ vertices in the subgraph $G[C] = (C, E \cap (C \times C))$ induced by C.

The maximum *s*-plex problem involves finding, for a fixed value of s, an *s*-plex of maximum cardinality among all possible *s*-plexes of a given graph. As indicated in Balasundaram et al. (2011), the maximum *s*-plex problem can be formulated as a binary linear program as follows:

$$
\begin{aligned}
&\max \ \omega_s(G) = \sum_{i \in V} x_i \\
&\text{s.t.} \sum_{j \in V \setminus (N(i) \cup \{i\})} x_j \leq (s-1)x_i + \bar{d}_i(1-x_i), \forall i \in V, \qquad (1) \\
&x_i \in \{0,1\}, \forall i \in V
\end{aligned}
$$

where x_i is the binary variable associated with vertex i, such that $x_i = 1$ if vertex i is in an *s*-splex, $x_i = 0$ otherwise. Also, $\bar{d}_i = |V \setminus N(i)| - 1$ denotes the degree of vertex i in the complement graph $\bar{G} = (V, \bar{E})$. Note that $i \notin N(i)$ by definition.

The *s*-plex concept was first introduced for graph-theoretic social network studies (Seidman and Foster, 1978). The decision version of the maximum *s*-plex problem with any fixed positive integer s is known to be NP-complete (Balasundaram et al., 2011). When s equals 1, the maximum *s*-plex problem reduces to the popular maximum clique problem, the decision version of which was among Karp's 21 NP-complete problems (Karp, 1972). The maximum *s*-plex problem is often referred to as a *clique relaxation* model (Pattillo et al., 2012, 2013b). Other clique relaxation models include *s*-defective clique (Yu et al., 2006), quasi-clique (Brunato et al., 2008; Pajouh et al., 2014; Pattillo et al., 2013a), and *k*-club (Bourjolly et al., 2000), which are defined by relaxing the edge number, the edge density, and the pairwise distance of vertices in an induced subgraph, respectively. In addition to studies of social networks, the maximum *s*-plex problem has also been investigated in other contexts (Berry et al., 2004; Boginski et al., 2014; Gibson et al., 2005). For instance, an interesting application of the maximum *s*-plex model was described in Boginski et al. (2014), where the maximum *s*-plex algorithm of Trukhanov et al. (2013) was used to find profitable diversified portfolios on the stock market.

Similar to a clique, an *s*-plex C has the heredity property, which means that every subset of vertices $C' \subset C$ remains an *s*-plex, i.e., the subgraph induced by C' always has the property of an *s*-plex (Trukhanov et al., 2013). The most successful combinatorial algorithms for *s*-plex essentially rely on the heredity property and a polynomial feasibility verification procedure. For example, a powerful exact algorithmic framework was introduced in Trukhanov et al. (2013) for detecting optimal hereditary structures

* Corresponding author at: LERIA, University of Angers, 2 bd Lavoisier, 49045 Angers, France
E-mail address: jin-kao.hao@univ-angers.fr (J.-K. Hao).

http://dx.doi.org/10.1016/j.cor.2017.05.005

(s-plex and s-defective clique), which is based on the maximum clique algorithm proposed in Östergård (2002). This algorithm performed well on the maximum s-plex problem for graphs in the 2nd DIMACS Challenge and popular large-scale social networks. Other exact algorithms for the s-plex problem include the following. A branch-and-cut algorithm was introduced in Balasundaram et al. (2011) based on a polyhedral study of the s-plex problem. Two branch-and-bound algorithms were presented in McClosky and Hicks (2012), which are based on popular exact algorithms for the maximum clique problem (Carraghan and Pardalos, 1990; Östergård, 2002). In Moser et al. (2012), exact combinatorial algorithms were investigated using methods from parameterized algorithmics. Finally, a parallel algorithm for listing all the maximal s-plexes was introduced in Wu and Pei (2007).

However, given the computational complexity of the maximum s-plex problem, any exact algorithm is expected to require an exponential computational time to determine the optimal solution in the general case. Thus, it is useful to investigate heuristic approaches, which aim to provide satisfactory solutions within an acceptable time frame, but without a provable optimal guarantee for the solutions obtained. However, our literature review found only two heuristics for the maximum s-plex problem (Gujjula et al., 2014; Miao and Balasundaram, 2012), which are based on the general GRASP method (Resende and Ribeiro, 2010). This situation contrasts sharply with the huge body of heuristics for the conventional maximum clique problem (Wu and Hao, 2015) and other clique relaxation problems (Pattillo et al., 2013b). We note that exact and heuristic approaches may complement each other, and together they can enlarge the classes of problem instances that can be solved effectively. Moreover, they can even be combined within a hybrid approach, as exemplified in Miao and Balasundaram (2012) where the GRASP heuristic was used to enhance the exact algorithm proposed in Balasundaram et al. (2011) to solve very large social network instances.

In this study, we aim to partially fill the gap in terms of heuristic methods for solving the maximum s-plex problem by introducing an effective heuristic approach. The main contributions of this study can be summarized as follows.

- From an algorithmic perspective, this is the first study to employ the tabu search metaheuristic (Glover and Laguna, 1997) to solve the maximum s-plex problem (Section 2). Thus, the proposed frequency-driven tabu search algorithm (FD-TS) integrates several original components. First, FD-TS jointly employs three dedicated move operators called *Add*, *Swap*, and *Press*, two of which (*Swap* and *Press*) are applied for the first time to the maximum s-plex problem. Second, we introduce a frequency-based mechanism for perturbation and constructing initial solutions, which is proven to be more effective than a random mechanism. We also apply a peeling procedure to dynamically reduce the graph with the best identified lower bound. Finally, specific design decisions are made in order to handle very large networks with thousands and even millions of vertices.
- From a computational perspective, our experimental results indicate that the proposed algorithm performs very well with both sparse and dense graphs (Section 4). For 47 very large networks from the SNAP collection and the 10th DIMACS Challenge benchmark set, our algorithm successfully obtained or improved the best-known results from previous studies for $s = 2, 3, 4, 5$. Our algorithm even proved the optimality of many instances for the first time using the peeling procedure. For 52 dense graphs from the collection used in the 2nd DIMACS Challenge, our algorithm also obtained or improved the best-known results for $s = 2, 3, 4, 5$. To comprehensively assess the performance of our algorithm, we compared FD-TS with several cutting edge algorithms, including the commercial CPLEX solver (version 12.6.1). Results of 48 additional graphs for the s-plex problem are also presented for the first time in the Appendix.

Algorithm 1. Main framework of frequency driven tabu search.

Input: Problem instance (G, s), predefined sample size q, maximum allowed iterations in tabu search L.
Output: The largest s-plex ever found

```
1  begin
2      C* ← ∅;                                /* the best solution found so far */
3      freq(v) ← 0 for all v ∈ V;   /* frequency count of vertex moves */
4      while the stopping condition is not met do
5          {C, freq} ← Init_Solution(G, s, freq, q);                 /* §2.4 */
6          {C, freq} ← Freq_Tabu_Search(G, s, C, freq, L);           /* §2.5 */
7          if |C| > |C*| then
8              C* ← C;
9              G ← Peel(G, s, |C*|);                                 /* §2.6 */
10             if |V| ≤ |C*| then
11                 return C* ;        /* return the best solution found */
12 end
13 return C*
```

The remainder of this paper is organized as follows. Section 2 presents the FD-TS algorithm. Section 3 discusses the implementation and complexity issues related to FD-TS. Section 4 presents the computational results obtained on benchmark instances and provides comparisons with state-of-the-art algorithms. In the final section, we give our conclusions and discuss future research.

2. FD-TS algorithm for the maximum *s*-plex problem

2.1. General procedure

The general scheme of the proposed FD-TS algorithm is shown in Algorithm 1. FD-TS starts from an initial feasible solution (s-plex) built using the *Init_Solution()* procedure (Section 2.4), before entering the main multi-neighborhood local search procedure, *Freq_Tabu_Search()*, to improve the initial solution (Section 2.5). A vector *freq*, which records the number of times each vertex is moved in the last round of the *Freq_Tabu_Search()* procedure, is initialized as a null vector (Algorithm 1, line 3). This vector is used by the *Init_Solution()* procedure as well as the perturbation method explained in Section 2.5.3. If the solution returned by tabu search is better than the current best solution C^*, C^* is updated (Algorithm 1, lines 7–8). The new lower bound $|C^*|$ is then given to the *Peel()* procedure (Section 2.6) to reduce the current graph (Algorithm 1, line 9). If *Peel()* returns a reduced subgraph with fewer vertices than $|C^*|$, then C^* must be an optimal solution and the overall algorithm stops. Otherwise, the algorithm enters a new round of search to build a new starting solution with *Init_Solution()*, before improving the new starting solution with *Freq_Tabu_Search()* and reducing the graph with *Peel()* if this is possible. The algorithm continues until a given stopping condition (e.g., a cut-off time limit) is met.

2.2. Preliminary definitions

Given $G = (V, E)$, $s \in \mathbb{Z}^+$, let $C \subseteq V$ be a subset of vertices and $N(v)$ the set of vertices adjacent to v. The following definitions are provided, which are useful for the description of our algorithm.

We say that C is a (feasible) solution or an s-plex if $\forall v \in C, |N(v) \cap C| \geq |C| - s$; otherwise, C is an infeasible solution (i.e., $\exists v \in C, |N(v) \cap C| < |C| - s$). For a vertex $v \in C$, we say that v is *saturated* (first introduced in Trukhanov et al. (2013)) if $|N(v) \cap C| = |C| - s$. If $|N(v) \cap C| < |C| - s$, v is *deficient*. Obviously, whenever a