A Declarative Framework for Maximal k-plex Enumeration Problems

Said Jabbour
CRIL - CNRS, Université d'Artois
France
jabbour@cril.fr

Nizar Mhadhbi
INSY2S
France
mhadhbi.nizar@gmail.com

Badran Raddaoui
Télécom SudParis, Institut Polytechnique de Paris
France
badran.raddaoui@telecom-sudparis.eu

Lakhdar Sais
CRIL - CNRS, Université d'Artois
France
sais@cril.fr

ABSTRACT

It is widely accepted that an ideal community in networks is the one whose structure is closest to a (maximal) clique. However, in most real-world graphs the clique model is too restrictive, as it requires complete pairwise interactions. More relaxed cohesive subgraph models were then studied. A k-plex is one of the arguably most studied pseudo-clique model. A k-plex of size n is a subgraph where any vertex is adjacent to at least $(n - k)$ vertices. Unfortunately, some maximal k-plexes, by involving irrelevant subgraphs, are far from designing meaningful communities in real-world networks. In this paper, we first introduce a novel variant of k-plex model, called *cohesive k-plex*, which is more appropriate for modeling closely-interacting communities. Then, we reduce the problem of enumerating maximal (cohesive) k-plexes in a graph to those of enumerating the models of a formula in propositional logic. Afterwards, to make our approach more efficient, we provide a decomposition technique that is particularly suitable for deriving smaller and independent sub-problems easy to resolve. Lastly, our extensive experiments on various real-world graphs demonstrate the efficiency of the proposed approach w.r.t state-of-the-art algorithms.

KEYWORDS

k-plex, Graph Enumeration, Community Detection, Propositional Satisfiability

ACM Reference Format:
Said Jabbour, Nizar Mhadhbi, Badran Raddaoui, and Lakhdar Sais. 2022. A Declarative Framework for Maximal k-plex Enumeration Problems. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Auckland, New Zealand, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

In social network analysis, detecting a large cohesive sub-graph is a fundamental and extensively studied topic with various applications in, e.g., sociology, bibliography, and biology [11, 30]. Clique is one of the ideal structures, widely used in the field of community discovery. A clique is a graph with an edge between any pair of vertices, which can be considered as the most cohesive class of graph. The computation of maximal cliques in graphs has been extensively studied, with a plethora of applications in various areas such as data mining, genomics and biochemistry [31], ad hoc wireless networks [15], and overlapping community discovery [23, 28], etc. Unfortunately, in most real-world graphs the clique structure is not widespread, as it requires complete pairwise interactions.

Several kinds of *pseudo-cliques* have been introduced as a relaxation of the clique and have been used to find overlapping subgroups in real networks. *k-plex*, one of the important relaxation of a clique, has gained ever-increasing popularity in recent years [2, 3, 38]. However, the number of k-plexes can be exponential in the size of a the input graph, and some of these k-plexes cannot be considered as meaningful communities since they are highly sparse or disconnected.

Declarative approaches, using propositional satisfiability (SAT) and constraint programming (CP), have been successfully applied to model and solve several pattern and graph mining problems [14, 16, 18, 20–22, 25]. Following this emergent research trend, in this paper, we first introduce a novel variant of the k-plex structure, named *cohesive k-plex*, to ensure the cohesiveness between all the vertices in the k-plex subgraph. Our intuition is that the counter part of the clique relaxation constraint is to maintain the shortest distance among vertices in the k-plex. This allows us to significantly reduce the number of irrelevant k-plexes in a given graph. Next, we propose a declarative approach to enumerate all maximal (cohesive) k-plexes in graphs. At the core of this novel approach is a SAT-based encoding which takes an undirected graph $\mathcal{G}$ and a positive value k as input and returns a propositional formula whose models correspond to the maximal (cohesive) k-plexes in $\mathcal{G}$. In this paper, we focus on cohesive k-plexes with low k values ($k = 1, 2, 3$)[1], since they produce clique relaxations found in real-life networks [10, 26]. Afterwards, even if our SAT-based encoding is polynomially bounded, it could become intractable in practice when dealing with very large graphs. To overcome the scalability issue, we harness a graph decomposition technique to generate many independent and smaller enumeration sub-problems. This allows an easy parallelization of the enumeration process of maximal (cohesive) k-plexes when dealing with big graphs.

Our contributions can be summarized as follows:

(1) To avoid the main drawbacks of the k-plex structure, a novel extension, called *cohesive* k-plex, is proposed. It is obtained

[1]Our approach works for general cases, i.e., $k \geq 4$.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Auckland, New Zealand.

thanks to additional constraints, that ensure connectivity and maintain the distance between vertices.
(2) We propose a SAT-based encoding of the problem of enumerating both maximal k-plexes and maximal cohesive k-plexes.
(3) We enhance the efficiency of our SAT-based encoding by providing a decomposition technique suitable for deriving smaller and independent enumeration sub-problems easy to resolve.
(4) The efficiency of our approach is extensively evaluated on commonly used real-world graphs.

2 TECHNICAL BACKGROUND

2.1 Propositional Logic and SAT Problem

A propositional language $\mathcal{L}$ is defined inductively using a finite set of boolean variables, the constants $\top$ (*true* or 1) and $\bot$ (*false* or 0) and the classical connectives $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$ as usual. Letters x, y, z, etc. denote the propositional variables. Propositional formulas of $\mathcal{L}$ are denoted by Φ, Ψ, etc. A literal is defined as a propositional variable (x) or its negation ($\neg x$). A propositional formula in conjunctive normal from (CNF, for short) is defined as a conjunction of clauses s.t. a *clause* is a disjunction of literals, i.e., $x_1 \vee \ldots \vee x_n$. For every propositional formula Φ from $\mathcal{L}$, $\mathcal{P}(\Phi)$ denotes the boolean symbols occurring in Φ. Given a propositional formula Φ, a Boolean interpretation $\mathcal{I}$ of Φ is a truth assignment of the set of variables in Φ, i.e., a total function from $\mathcal{P}(\Phi)$ to $\{0, 1\}$. A *model* of a formula Φ is a Boolean interpretation $\mathcal{I}$ that satisfies Φ. A formula Φ is satisfiable if there exists a model of Φ. We write $\mathcal{M}(\Phi)$ for the models of Φ. Propositional satisfiability (SAT, for short) is the NP-complete problem that verifies the (non) existence of a model for a CNF formula. On the one hand, the SAT model enumeration problem considered in this paper generalizes the classical SAT problem to deal with the computation of all the models of the formula. On the other hand, SAT has been applied in various fields such as community detection, and data mining [18, 20].

2.2 Cliques and k-plexes

An undirected graph is formally defined as a pair $\mathcal{G} = (V, E)$ where V is a set of vertices and $E \subseteq \{\{u, v\} \mid u, v \in V\}$ is a set of edges. The density of $\mathcal{G}$ is defined as $\frac{2|E|}{|V|(|V|-1)}$. Given a vertex $u \in V$, the set of adjacent (neighbor) vertices of u is denoted by $\Gamma(u) = \{v \mid (u, v) \in E\}$. The degree of u in $\mathcal{G}$ is $deg_{\mathcal{G}}(u) = |\Gamma(u)|$. The length of the shortest path between two vertices $u, v \in V$ in $\mathcal{G}$ is called the *distance* between u and v, denoted by $d_{\mathcal{G}}(u, v)$. Given an undirected graph $\mathcal{G} = (V, E)$, we define the subgraph induced by $S \subseteq V$ as $\mathcal{G}[S] = (S, E')$ s.t. $E' \subseteq E$. Given a graph $\mathcal{G} = (V, E)$, a *clique* in $\mathcal{G}$ is a complete subgraph $\mathcal{G}[C]$, i.e., $\forall u \in C, deg_{\mathcal{G}[C]}(u) = |C| - 1$. A clique $\mathcal{G}[C]$ is called a *maximal clique* in $\mathcal{G}$ if there exists no clique $\mathcal{G}[C']$ such that $C \subset C'$. The enumeration of maximal cliques is very important as it appears at the core of many real-world networks analyses. In this work, we focus on the enumeration of k-plexes, a widely studied structure of pseudo-cliques [9], defined as follows:

Definition 2.1 (k-plex). Let $\mathcal{G} = (V, E)$ be an undirected graph. Then, a k-plex in $\mathcal{G}$, where k is a positive integer, is a subgraph $\mathcal{G}[P]$ such that $\forall u \in P, deg_{\mathcal{G}[P]}(u) \geq |P| - k$.

Clearly, a 1-plex is a clique, and each vertex of a 2-plex can miss one edge. Obviously, any subgraph of a k-plex is a k-plex, and a k-plex is also a $k+1$-plex.

Definition 2.2 (Maximal k-plex). Let $\mathcal{G} = (V, E)$ be an undirected graph. A k-plex $\mathcal{G}[P]$ is maximal if there is no $V \supseteq P' \supset P$ such that $\mathcal{G}[P']$ is a k-plex.

In [33], Seidman states that a k-plex is a type of cohesive subgraph where any vertex may miss k neighbors. However, the k-plex subgraph may be disconnected, and it might have multiple connected components. For instance, in Example 2.4, the 3-plex $\mathcal{G} = (\{1, 5, 6, 7\}, \{(1, 6), (5, 7)\})$ forms a graph with two connected components (see Figure 1). Hence, such a disconnected 3-plex is irrelevant and far from being a real community.

Now, on top of the notion of k-plex, we add an *edge connectivity* constraint, that is, any two vertices in the k-plex are reachable from each other through a path. This gives rise to the notion of maximal *connected* k-plex defined as follows.

Definition 2.3 (Maximal Connected k-plex). A graph $\mathcal{G} = (V, E)$ is a maximal connected k-plex if $\mathcal{G}$ is a maximal k-plex and $\mathcal{G}$ is connected.

In the following, when there is no ambiguity, we call a maximal connected k-plex simply a maximal k-plex. Next, we show that despite the edge connectivity requirement, maximal k-plexes might involve irrelevant subgraphs, making them inappropriate for expressing real-world communities.

Example 2.4. Consider the graph depicted in Figure 1. Suppose we are searching for maximal 4-plexes. Then, we have:

$\{1, 2, 3, 4, 5\}$ $\{1, 2, 3, 4, 6, 7\}$ $\{1, 2, 3, 5, 7\}$ $\{1, 3, 4, 5, 6\}$
$\{1, 2, 4, 5, 6, 7\}$ $\{1, 3, 5, 6, 7\}$ $\{2, 3, 4, 5, 6, 7\}$

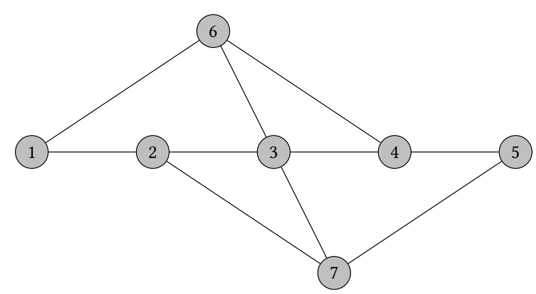

Figure 1: An illustrative example

As we can observe, the maximal 4-plexes $\{1, 2, 3, 4, 5\}$, $\{1, 3, 5, 6, 7\}$, and $\{1, 2, 4, 5, 6, 7\}$ represent simple paths of 5 and 6 vertices, respectively, which are far from expressing real communities. In addition, these three maximal 4-plexes correspond to those with the smallest density ($\frac{8}{20} = 0.4$). All the remaining maximal 4-plexes admit higher densities, i.e., greater than 0.5.

The graph depicted in Figure 1 shows clearly that some maximal k-plexes are not appropriate to model real-world community structures. In fact, while a k-plex, a relaxed version of a clique, captures a larger set of communities, there is a negative counter part as such relaxation does not preserve the shortest paths between vertices.

In what follows, we extend the notion of k-plex by adding a new constraint on the distance between vertices. Intuitively, we require that the distance among any pair of vertices in the k-plex subgraph is equal to the shortest path between these two vertices in the original graph. Given two vertices u and v, our main idea is that u and v belong to the same cluster, if u and v are close together w.r.t. their distance. More formally,

Definition 2.5 (Maximal Cohesive k-plex). Let $\mathcal{G} = (V, E)$ be an undirected graph and $\mathcal{G}[P]$ a maximal k-plex in $\mathcal{G}$. Then, $\mathcal{G}[P]$ is a maximal cohesive k-plex iff for all $u, v \in P$, $d_{\mathcal{G}[P]}(u, v) = d_{\mathcal{G}}(u, v)$.

Unlike k-plexes, a maximal cohesive k-plex aims to maintain the pairwise distance between vertices as in the original graph $\mathcal{G}$, and consequently it captures real-world communities.

Example 2.6. Let us consider again the undirected graph depicted in Figure 1. Then, the set of maximal cohesive 4-plexes are:

$$\{1, 2, 3, 4, 6, 7\} \quad \{1, 2, 3, 5, 7\} \quad \{1, 3, 4, 5, 6\} \quad \{2, 3, 4, 5, 6, 7\}$$

Clearly, among the 7 maximal 4-plexes (Example 2.4), the above 4 maximal cohesive 4-plexes correspond to those with higher density.

Definition 2.5 imposes that if two vertices are in a k-plex then they are reachable from each other through a shortest path according to $\mathcal{G}$. This allows circumventing the computation of irrelevant k-plex subgraphs. Moreover, maximal cohesive k-plexes are appropriate to capture the cohesiveness and high-density in real-world communities.

In this work, we investigate the problem of enumerating maximal (cohesive) k-plexes of specified size at least $\alpha > 2$, which is formulated as follows.

Problem Definition. Maximal **C**ohesive k-**P**lex **E**numeration problem (MCkPE, for short):
Input: $\mathcal{G}$ an undirected graph and k a positive integer.
Output: The set of maximal (cohesive) k-plexes in $\mathcal{G}$.

As illustrated in the previous example, the number of maximal cohesive k-plexes is smaller than those of maximal k-plexes. Obviously, such reduction of the output induces practical performance improvements. Unfortunately, in the worst case the MCkPE problem is intractable. In this paper, we address this limitation by providing a practical SAT-based approach to solve efficiently the MCkPE problem.

3 SAT-BASED FRAMEWORK FOR MCKPE

This section outlines our declarative approach to solve the MCkPE problem. Given an undirected graph $\mathcal{G}$, MCkPE is modeled as a logical formula whose models are the maximal cohesive k-plexes in $\mathcal{G}$. Specifically, MCkPE is reduced to the computation of the set of models in propositional logic.

Note that the separation of the modeling step from the solving step has two important benefits. It first offers an easy way to integrate a possible evolution in the problem specification, by simply incorporating new logical constraints. Secondly, the solving step can be continuously enhanced by considering the last advances in SAT solvers and model enumeration algorithms.

A recent breakthrough in the efficiency of state-of-the-art SAT solving technology opens an avenue for encoding various real-world problems to propositional logic. However, it is important to note that the problem modeling might have a substantial impact on the efficiency of the solving process. Hence, the challenge is to provide the most succinct and efficient SAT encoding, while ensuring correctness and completeness. This requires a sagacious selection strategy of boolean variables and constraints as well as their formulation in CNF.

Prior to the presentation of our SAT-based encoding of MCkPE, we first introduce an important property of the k-plex structure. An important observation that can be made is that a clique ensures the smallest distance between any pair of vertices. The following property expresses an upper bound distance between any two vertices in a k-plex:

Property 1. *Let $\mathcal{G} = (V, E)$ be a k-plex and $k \geq 1$ a positive integer. If $\mathcal{G}$ is connected, then $\forall u, v \in V, d_{\mathcal{G}}(u, v) \leq k$.*

Proof. Assume that there exist two vertices $u, v \in V$ such that $d_{\mathcal{G}}(u, v) \geq k + 1$. Then, u cannot be connected to k vertices. As every vertex of a k-plex is not connected to at most $k - 1$, $\mathcal{G}$ is not a k-plex. □

3.1 A SAT-based Encoding of MCkPE

Now, let us introduce our practical SAT-based encoding of MCkPE on a given graph $\mathcal{G} = (V, E)$, which allows for using off-the-shelf SAT solvers to enumerate maximal (cohesive) k-plexes induced from $\mathcal{G}$. For this, we present the following propositional variables and logical constraints:

Variables. We use propositional variables to indicate the inclusion of a vertex u in the k-plex. For each vertex $u \in V$, we associate a propositional variable x_u such that $x_u = 1$ encodes that u belongs to the candidate k-plex subgraph.

Constraints. In the sequel, we present the different Boolean constraints allowing us to derive the CNF formula that encodes MCkPE.

k-plex relaxation constraint. The k-plex constraint is derived from the k-plex definition (Definition 2.1) expressing that each vertex is not connected to at most $k - 1$ vertices.

$$\bigwedge_{u \in V} (x_u \rightarrow \sum_{v \in V \mid 1 < d_{\mathcal{G}}(u,v) \leq k} x_v < k) \tag{1}$$

Intuitively, a vertex u is involved in a k-plex iff the number of vertices that are not neighbor of u does not exceed $(k - 1)$. Constraint (1) is a conjunction of the so-called *conditional cardinality constraints* of the from $y \rightarrow \sum_{i=1}^{n} x_i \leq k$. It generalizes the cardinality constraints that naturally arise in many SAT encodings of real-world problems, including pattern mining, product configuration and community discovery. Several encodings have been designed, translating cardinality constraints into CNF (e.g. Totalizer encoding [1], Sequential counter [34], Adder [12], pigeon-hole based encoding [19]). More recently, in [4], the authors have shown how to extend such encodings to conditional cardinality constraints in an efficient way.

k-plex filtering constraint. The second constraint, called *k-plex filtering constraint*, follows from Property 1. It allows to circumscribe the potential vertices of the k-plex by eliminating those which cannot be part of the candidate k-plex. Such filtering is achieved by expressing that for any vertex u in the k-plex $\mathcal{G}$, every vertex v at distance greater or equal to $(k+1)$ is not included in $\mathcal{G}$.

$$\bigwedge_{u,v \in V \mid d_{\mathcal{G}}(u,v) > k} (\neg x_u \vee \neg x_v) \tag{2}$$

Minimal size constraint. Now, the lower bound of the size of a k-plex can be modeled as the next cardinality constraint:

$$\sum_{u \in V} x_u \geq \alpha \tag{3}$$

In this paper, we consider graphs without isolated vertices and edges (i.e., $\alpha > 2$). Indeed, these particular subgraphs are straightforward to be found and eliminated during the preprocessing step.

Maximality constraint. The following constraint provides the condition under which a k-plex is maximal.

$$\begin{aligned}\bigwedge_{u \in V} [\neg x_u \rightarrow (&\sum_{v \in V \mid 1 < d_{\mathcal{G}}(u,v) \leq k} x_v \geq k) \vee & [a]\\ &(\bigvee_{v \in V \mid d_{\mathcal{G}}(u,v) > k} x_v) \vee & [b]\\ (\bigvee_{v \in V \mid 1 < d_{\mathcal{G}}(u,v) \leq k} &(x_v \wedge \sum_{w \in V \mid 1 < d_{\mathcal{G}}(v,w) \leq k} x_w = k-1))] & [c]\end{aligned} \tag{4}$$

Intuitively, Constraint (4) expresses the fact that a vertex u is not involved in the maximal k-plex $\mathcal{G}$ if $\mathcal{G}$ contains:

(1) k vertices not connected to u (i.e., sub-formula [a]). Then, u cannot be included as it violates the k-plex definition,
(2) there exists a vertex v at distance $(k+1)$ from u (i.e., sub-formula [b]). Adding u to the k-plex would result in the violation of Property 1.
(3) a vertex v not connected to u such that there exists $(k-1)$ vertices not connected to v (i.e., sub-formula [c]). Adding u to the k-plex would increase the number of vertices not connected to v up to k.

Next, we illustrate the maximality constraint using Example 3.1.

Example 3.1. Let us again consider the graph depicted in Figure 1, and $k = 2$. If we consider the vertex 1, the maximality constraint gives the following formula:

$$\begin{aligned}\neg x_1 \rightarrow [&(x_3 + x_4 + x_7 \geq 2) \vee & [a]\\ &(x_5) \vee & [b]\\ &(x_3 \wedge (x_1 + x_5 = 1)) \vee (x_4 \wedge (x_1 + x_2 + x_7 = 1) \vee &\\ &(x_7 \wedge (x_1 + x_4 + x_6 = 1))] & [c]\end{aligned}$$

Next, we can improve our approach by rewriting Constraint (4) of the SAT-based encoding of MCkPE as follows:

$$\begin{aligned}&\bigwedge_{u \in V} (\neg x_u \rightarrow z_u \vee \bigvee_{v \in V \mid d_{\mathcal{G}}(u,v) > k} x_v \vee \bigvee_{v \in V \mid 1 < d_{\mathcal{G}}(u,v) \leq k} (x_v \wedge y_v)) \quad \wedge\\ &\bigwedge_{u \in V} (z_u \rightarrow \sum_{v \in V \mid 1 < d_{\mathcal{G}}(u,v) \leq k} x_v \geq k) \quad \wedge\\ &\bigwedge_{v \in V} (y_v \rightarrow \sum_{w \in V \mid 1 < d_{\mathcal{G}}(v,w) \leq k} x_w \geq k-1)\end{aligned} \tag{5}$$

Notice that Constraint (5) is obtained from the formula (4) by respectively substituting the sub-formulas $\sum_{v \in V \mid 1 < d_{\mathcal{G}}(u,v) \leq k} x_v \geq k$ and $\sum_{w \in V \mid 1 < d_{\mathcal{G}}(v,w) \leq k} x_w \geq k-1$ with fresh propositional variables z_u and y_v. As the two sub-formulas appear with positive polarities [29], they can simply be defined using logical implications. Moreover, the second sub-formula can be represented by the conjunction of two linear inequalities. One of them is subsumed by the k-plex constraint. This new formulation can be efficiently translated into a CNF using the encodings of conditional cardinality constraints proposed in [4].

Proposition 3.2. *Let $\mathcal{G} = (V, E)$ be a graph and $\Phi = (1) \wedge (2) \wedge (3) \wedge (5)$. Then, there exists a one-to-one mapping between the models of Φ and the maximal k-plexes of size at least α induced from $\mathcal{G}$.*

The above proposition shows the formula modeling the maximal k-plex enumeration problem whose models correspond to the maximal k-plexes of size at least α. The correctness and completeness of our SAT-based encoding follows from the logical constraints introduced previously. In the particular case of a clique (or 1-plex), our practical SAT-based encoding of MCkPE can be simplified to the following CNF:

$$\bigwedge_{u \in V} (x_u \leftrightarrow \bigwedge_{v \in V,\ (u,v) \notin E} \neg x_v) \tag{6}$$

In fact, by setting k to 1, Constraint (1) is trivially true, i.e., the right hand side of the implication is reduced to the true statement. The $(\rightarrow)$ implication of Constraint (6) models the distance constraint. Finally, as the sub-formulas $[a]$ and $[c]$ are trivially false, the maximality constraint corresponds to the $(\leftarrow)$ implication of Constraint (6).

Distance constraint. What remains is the encoding of the distance constraint allowing us to complete our SAT-based encoding of MCkPE. Let us recall that a k-plex is cohesive if it preserves the distance between vertices. This requirement is expressed via the next distance constraint defined as follows:

$$\bigwedge_{u,v \in V} (x_u \wedge x_v \rightarrow \bigvee_{w \in V,\ d_{\mathcal{G}}(w,v)=1,\ d_{\mathcal{G}}(u,v)=d_{\mathcal{G}}(u,w)+1} x_w) \tag{7}$$

Intuitively, Constraint (7) requires that if two vertices u and v are in the k-plex (i.e., x_u and x_v assigned *true*) such that $d_{\mathcal{G}}(u,v) = i$, then there exists a vertex w at distance 1 and $(i-1)$ from v and u, respectively. That is, u and v have to be reachable from each other through a shortest path in $\mathcal{G}$. Iteratively, Constraint (7) expresses the existence of a short path between u and v. Note that the number of clauses obtained from this constraint is bounded by n^2 with n is the number of vertices in the graph $\mathcal{G}$.

The following proposition shows the SAT-based encoding of the maximal cohesive k-plexes enumeration problem.

Proposition 3.3. *Let $\mathcal{G} = (V, E)$ be a graph and $\Phi = (1) \wedge (2) \wedge (3) \wedge (5) \wedge (7)$. Then, there exists a one-to-one mapping between the models of Φ and the maximal cohesive k-plexes of size at least α induced from $\mathcal{G}$.*

The distance constraint allows to avoid the irrelevant subgraphs mentioned previously (see Example 2.4). The following proposition highlights its importance for defining more cohesive and particularly connected k-plexes.

Proposition 3.4. *For a given graph $\mathcal{G}$, the k-plex relaxation constraint together with the distance constraint implies the connectivity constraint.*

Proof. Let $\mathcal{G}'$ be a maximal cohesive k-plex in $\mathcal{G}$. Consider two vertices u and v in $\mathcal{G}'$. Now, assume that u and v are not connected in $\mathcal{G}'$, i.e., $d_{\mathcal{G}'}(u,v) = \infty$. Then, due to the distance constraint u and v are also not connected in $\mathcal{G}$, i.e., $d_{\mathcal{G}}(u,v) = d_{\mathcal{G}'}(u,v) = \infty$. Now, the relaxation constraint implies that all vertices at distance at least $k+1$ from u cannot be connected to u, which contradicts the initial hypothesis that u and v are in $\mathcal{G}'$. □

As mentioned previously, the distance constraint ensures that the k-plex is connected. Now, if one needs the maximal connected k-plexes without preserving the distance (i.e. without Constraint 7), we should incorporate the following constraint in our SAT-based encoding.

Connectivity constraint. Let us recall that if $\alpha \geq 2k-1$, then any maximal k-plex is connected [10]. However, this result does not hold for $\alpha < 2k-1$. In what follows, we show how to ensure the existence of a path between every pair of vertices for $k \leq 3$. Obviously, for $k=1$ such constraint is useless; and, for $k=2$, the k-plex is connected as $\alpha \geq 3$.
We now present two constraints allowing to ensure the connectivity requirement when $k=3$. This is especially important in the case where $3 \leq \alpha \leq 4$, since many communities in real graphs tend to be small in size [37]. Given a 3-plex $\mathcal{G} = (V,E)$, Constraint (8) ensures that for any vertex $u \in V$, there exists at least a vertex $v \in \Gamma(u)$ s.t. $v \in V$. Also, Constraint (9) enforces that for each edge $(u,v) \in E$, there exists at least a vertex $w \in V$ such that $w \in \Gamma(u)$ or $w \in \Gamma(v)$. Obviously, for any vertex $t \notin \{u,v,w\} \subseteq V$, Constraints (8) and (9) ensure that t is connected to at least one vertex from $\{u,v,w\}$. Hence, $\mathcal{G}$ is a connected k-plex.

$$\bigwedge_{u\in V} (x_u \rightarrow \bigvee_{v\in\Gamma(u)} x_v) \tag{8}$$

$$\bigwedge_{(u,v)\in E} x_u \wedge x_v \rightarrow \bigvee_{w\in\Gamma(u)\setminus\{v\}} x_w \vee \bigvee_{w\in\Gamma(v)\setminus\{u\}} x_w \tag{9}$$

Other pruning constraints. Additionally, in order to prune the search tree new constraints can be added to our SAT-based encoding. More specifically, the first constraint we consider relies on the fact that for a k-plex $\mathcal{G}$ of size α, it is easy to see that $(\alpha-k)$ vertices adjacent to vertex u are in $\mathcal{G}$.

$$\bigwedge_{u\in V} (x_u \rightarrow \sum_{v\in V,\ (u,v)\in E} x_v \geq \alpha - k) \tag{10}$$

That is, the input graph can be simplified by removing all vertices with degrees less than $\alpha - k$. Another less obvious, yet essential cut pointed out in [10], is that for each two vertices u and v in a k-plex $\mathcal{G}$, u and v have to share $\alpha - 2k + 2$ common neighbors in $\mathcal{G}$. Formally, this can be written as:

$$\bigwedge_{u,v\in V} (x_u \wedge x_v \rightarrow \sum_{w\in V,\ (u,w),(v,w)\in E} x_w \geq \alpha - 2k + 2) \tag{11}$$

3.2 Decomposition-based SAT Encoding for MCkPE

Let us remark that our practical SAT-based encoding of MCkPE is polynomial in the size of the graph. Whatever the well-known encoding of the conditional cardinality constraints (e.g., [4, 12, 34]), the number of propositional variables and clauses is bounded by $O(n^3)$ where n is the number of vertices in the original graph. Unfortunately, on very large graphs, such complexity, even if it is polynomially bounded, tends to be intractable in practice. To overcome the scalability issue, we propose a *decomposition technique* improving the performances by avoiding the generation of large CNF formulas. More specifically, let us recall the Shannon's decomposition theorem of a propositional formula stating that for a formula Φ and a variable x_u, the models of Φ can be decomposed into those containing x_u (i.e., $\Phi \wedge x_u$) and those containing $\neg x_u$ (i.e., $\Phi \wedge \neg x_u$). By generalizing this principle over the set of variables $\{x_{u_1}, \ldots, x_{u_n}\}$ in Φ, the set of models of Φ is then the union of the models of Φ_{u_i} $(1 \leq i \leq n)$ where $\Phi_{u_i} = \Phi \wedge \Psi_{u_i}$ s.t. Φ is the formula encoding MCkPE and $\Psi_{u_i} = (\bigwedge_{1\leq j<i} \neg x_{u_j}) \wedge x_{u_i}$ is the guiding path of the decomposition.

Algorithm 1, coined **SAPE** (**S**AT based m**A**ximal (cohesive) k-**P**lexes **E**numeration), describes the pseudo-code of our practical SAT modeling and solving approach of MCkPE. What is important to note is that we can avoid the generation of the formula Φ encoding the whole graph $\mathcal{G} = (V,E)$. Indeed, the formula Φ_{u_i} can be obtained differently by adding conjunctively Ψ_{u_i} to the formula encoding MCkPE on the subgraph $\mathcal{G}'_{u_i} = (V', E')$ where $V' = \{v \mid d_{\mathcal{G}'_{u_i}}(u_i, v) \leq k\}$ and $E' = \{(v,w) \in E \mid v, w \in V'\}$ (lines 3 and 4 of Algorithm 1). The set of models are collected in line 7 via the function enumerateModels [17]. Each model is a maximal cohesive k-plex of size at least α in the original graph $\mathcal{G}$ (line 9 of Algorithm 1).

Algorithm 1: SAT based m**A**ximal cohesive k-**P**lexes **E**numeration (SAPE)

Data: $\mathcal{G} = (V = \{u_1, \ldots, u_n\}, E)$: a graph, $k \geq 1$ and $\alpha \geq 3$: two positive integers
Result: S: the set of all maximal cohesive k-plexes of size at least α

```
1 S ← ∅;
2 for i = 1 to n do
3     V' ← {v | d_{G'_{u_i}}(u_i, v) ≤ k};
4     E' ← {(v, w) ∈ E | v, w ∈ V'};
5     G'_{u_i} ← G(V', E');
6     Φ_{u_i} ← max_kPlex(G'_{u_i}, k) ∧ Ψ_{u_i};
7     S ← S ∪ enumerateModels(Φ_{u_i});
8 end
9 return S;
```

4 EXPERIMENTAL EVALUATION

In this section, we performed intensive experiments to evaluate our proposed approach. Algorithm 1 is implemented in C++ and used a modified MiniSAT as backend SAT solver for model enumeration[2]. For our decomposition technique, we consider the vertices of the input graph in ascending order w.r.t. the function f that associates for each vertex u the number of vertices at distance at most k from u. In fact, our goal is to start with easier regions of the input graph. Note that considering f in descending order is not efficient in practice. All experiments have been conducted on

[2]MiniSAT is a standard backtrack search algorithm for solving SAT problems: http://minisat.se/

Intel Xeon 3.30GHz processor with 64Gb memory on Linux CentOS machine. The cut off time was set to 2 hours for each run of an algorithm on a dataset; memory-out was set to 20 Gb for each such run. We also use the symbol (-) to mention that the method is not able to scale on the graph under the time limit. The implementation is available from https://github.com/anonyme971/k-plex. We conduct experiments over several real-world graphs to assess the performance of our declarative framework for computing all maximal k-plexes, maximal exact k-plexes[3], maximal cohesive k-plexes, and maximal cliques. The datasets, which are downloaded from the SNAP [24] and the network data repositories [32], represent different real-world applications (web networks, collaboration networks, social networks). The different characteristics of these datasets are given in Table 1. We also note that the reported runtime in all the experiments is in seconds.

Graph	**|V|**	**|E|**
Bio-CE	2 617	2 985
Bio-CE-Gt	924	3 239
Ca-Gr	5 242	14 496
Bio-Dmela	7 393	25 569
Ca-Hep	9 877	25 998
Gnutella	10 876	39 994
As-Caida	26 475	53 381
Road-Luxemb	114 599	119 666
Road-US-48	126 146	161 905
Road-US	129 164	165 435
Soc-Gemsec	47 500	222 887
Amazon	334 863	925 872
DBLP	317 080	1 049 866
Road-Pa	1 088 092	1 541 898
Road-Belgium	1 441 295	1 549 970
Road-Tx	1 379 917	1 921 660
Amazon0505	410 236	2 439 436
Road-Asia	11 950 757	12 711 603

Table 1: Summary of real-world datasets

We perform two kinds of experiments. The first one aims to compare the performances of our SAT-based approach for computing maximal k-plexes, against the state-of-the-art algorithms. In the second, we evaluate our declarative method to assess its performances on several variant of k-plex, including maximal exact k-plexes, maximal cohesive k-plexes and the particular case of maximal cliques. In this work, we only compared the performance since all algorithms produce the same output.

4.1 Maximal k-plexes Enumeration

This subsection shows the empirical evaluation of our SAPE method against the best known algorithms, namely D2K[4] [10] and GP [35], for enumerating the set of all maximal k-plexes in graphs. Since the requirement for D2K algorithm is that the size threshold α is at least $2k - 1$, all experiments are done by setting the value of $\alpha = 5, 10, 20$, following the work of [10]. For our algorithm, we also report in Table 2 the total number of enumerated maximal k-plexes (in parenthesis) for each benchmark instance.

Clearly, for all algorithms the running time is strongly related to the number of maximal k-plexes. Table 2 shows that our algorithm is the second-fastest method overall, next to D2K. It is also worthy pointing out that for several network datasets, our SAPE algorithm achieves competitive running time against D2K. For instance, with the different values of α, SAPE spends less than 0.5 second and its performance to enumerate the maximal 2-plexes is competitive with D2K on `Ca-Hep`, `Ca-Gr`, and `Gnutella` datasets. Interestingly enough, for the maximal 2-plex detection, SAPE is faster than D2K on `Road-US`, `Road-US-48`, `Road-Luxemb` and `Road-Belgium` graphs with the different values of α. In addition, when $k = 3$ and $\alpha \geq 10$, SAPE is faster than D2K on `Bio-CE`, `Amazon` and `Ca-Hep` graphs. It can also be observed that on runtime, SAPE is comparable to D2K on all cases except the three networks `Amazon0505`, `DBLP` and `As-Caida`. Nevertheless, we stress that D2K computes maximal k-plexes of bounded diameters. Indeed, the requirement for the D2K algorithm is that the diameter of any returned maximal k-plex is at most 2. Instead, SAPE and GP algorithms compute the set of all maximal k-plexes of any diameter.

As is apparent, SAPE works better than GP in almost cases for $\alpha = 5$ (e.g., 16/18 graphs for maximal 2-plexes). Interestingly enough, our approach outperforms GP for all datasets with $\alpha = 10, 20$ for $k = 2, 3$. For instance, on `Ca-Hep`, `Gnutella`, `Ca-Gr`, `Bio-Ce-Gt`, `Road-US`, `Road-Pa`, `Road-Tx`, `Soc-Gemsec`, and `Road-Belgium` graphs, SAPE computes all the maximal 2-plexes under 9.19 seconds, while GP algorithm takes more than 49 seconds to solve these datasets. Moreover, SAPE is at least 41 times faster than GP for enumerating all maximal 2-plexes on the `Road-Asia` dataset with more than 11 millions of vertices and edges. In addition, SAPE is at least 3 times faster than GP to compute all maximal 2-plexes on `DBLP` dataset. For 3-plex computation with $\alpha \geq 10$, SAPE performs best on all graphs.

Overall, the proposed SAT-based method is more efficient than the dedicated approach GP. Moreover, despite the efficiency of D2K, our declarative method is highly flexible. In other words, SAPE is very suitable for adding new user-specified constraints. In fact, in some applications, users may only be interested in some specific k-plexes, i.e., k-plexes of bounded size, k-plexes that (do not) contain vertices related to a user query, etc. Next, we illustrate the behavior of our algorithm for computing maximal k-plexes of size n where the degree of each vertex is exactly $n - k$.

4.2 Maximal Exact k-plexes Enumeration

In order to show the high flexibility of our SAT-based approach, this experiment is devoted to the enumeration of a particular kind of k-plex subgraph, which we called *maximal exact k-plex*. Without modifying the original code, this subgraph structure can be computed easily by replacing Constraint (1) in the previous SAT-based encoding with the following formula:

$$\bigwedge_{u \in V} (x_u \rightarrow \sum_{v \in V \mid 1 < d_{\mathcal{G}}(u,v) \leq k} x_v = k - 1)$$

[3]A maximal exact k-plex is defined as a maximal (w.r.t. set inclusion) set of vertices where each one is connected to all others except exactly $(k-1)$ vertices, i.e., $\forall u \in V$, $|\Gamma(u)| = |V| - k$.

[4]In [10], the authors have shown that their D2K algorithm is more efficient than the one proposed in [9], which in turn is shown more efficient than EnumIncExc [3].

Graph	**2-plexes**									**3-plexes**								
	$\alpha = 5$			$\alpha = 10$			$\alpha = 20$			$\alpha = 5$			$\alpha = 10$			$\alpha = 20$		
	SAPE	GP	D2K	SPAE	GP	D2K	SAPE	GP	D2K	SAPE	GP	D2K	SAPE	GP	D2K	SAPE	GP	D2K
Bio-CE	**0.01** (1)	1.58	**0.01**	**0.01** (0)	0.23	0.03	**0.01** (0)	0.17	**0.01**	**0.01** (161)	3.03	**0.01**	**0.01** (0)	2.22	**0.01**	**0.01** (0)	1.61	**0.01**
Bio-CE-Gt	1.49 (14322)	6.78	**0.03**	**0.01** (0)	9.99	**0.01**	**0.01** (0)	3.48	**0.01**	1893.90 (718483)	55.92	**6.71**	0.19 (473)	50.24	**0.02**	**0.01** (0)	56.78	**0.01**
Ca-Gr	0.20 (4057)	7.60	**0.01**	0.05 (377)	6.59	**0.01**	**0.01** (118)	14.67	**0.01**	131.13 (667335)	52.55	**2.09**	0.56 (13352)	45.61	**0.05**	**0.01** (1568)	45.33	**0.01**
Bio-Dmela	1.79 (5897)	15.89	**0.08**	**0.01** (0)	21.13	0.05	**0.01** (0)	20.91	0.04	4411.59 (1286244)	724.66	**22.69**	**0.01** (0)	726.88	**0.01**	**0.01** (0)	599.62	**0.01**
Ca-Hep	0.47 (5894)	15.93	**0.04**	**0.01** (5)	10.65	**0.01**	**0.01** (3)	71.28	**0.01**	148.46 (502685)	91.68	**3.36**	**0.01** (5)	83.58	**0.01**	**0.01** (3)	71.28	**0.01**
Gnutella	0.09 (122)	23.89	**0.02**	**0.01** (0)	10.18	0.02	**0.01** (0)	14.01	0.02	73.59 (105763)	217.58	**6.96**	**0.01** (0)	229.61	0.09	**0.01** (0)	190.05	0.02
As-Caida	1038.92 (364674)	300.81	**4.12**	52.09 (23314)	239.34	**0.67**	0.48 (0)	207.40	**0.03**	–	–	–	–	–	**42.01**	0.19 (0)	–	**0.01**
Road-Luxemb	**0.01** (0)	3.62	0.22	**0.01** (0)	1.62	0.11	**0.01** (0)	1.50	0.11	0.80 (287)	5.02	**0.00**	**0.10** (0)	1.73	**0.10**	**0.01** (0)	1.50	0.11
Road-US-48	**0.19** (2)	7.45	0.37	**0.09** (0)	3.08	0.32	**0.09** (0)	1.88	0.13	1.49 (4992)	13.57	**0.29**	**0.09** (0)	3.95	0.13	**0.09** (0)	2.00	0.14
Road-US	**0.09** (4)	11.11	0.12	**0.09** (0)	3.21	0.11	**0.09** (0)	1.97	0.15	2.09 (5089)	13.51	**0.41**	**0.09** (0)	4.18	0.32	**0.09** (0)	1.97	0.37
Soc-Gemsec	9.19 (50812)	46.20	**5.07**	0.70 (119)	43.10	**0.09**	0.09 (0)	35.49	**0.01**	2626.01 (2998194)	1878.14	**7.03**	4.68 (1230)	1624.28	**2.01**	0.09 (0)	1504.45	**0.01**
Amazon	18.08 (355033)	59.61	**1.93**	**0.65** (0)	56.02	0.69	**0.36** (0)	48.19	0.72	–	1471.43	**343.14**	**0.78** (0)	85.71	1.37	**0.39** (0)	32.74	0.72
DBLP	119.39 (458915)	228.84	**3.45**	10.96 (20093)	173.71	**0.86**	3.66 (5049)	157.15	**0.78**	–	–	**1397.10**	1819.17 (3533545)	–	**20.13**	546.08 (2141776)	–	**14.73**
Road-Pa	3.74 (57)	40.91	**0.85**	0.90 (0)	27.97	**0.09**	0.85 (0)	21.24	**0.05**	17.64 (125649)	61.14	**13.12**	0.88 (0)	32.80	**0.12**	0.81 (0)	19.17	**0.01**
Road-Belgium	**1.09** (1)	42.99	2.07	**1.09** (0)	38.95	2.14	**1.09** (0)	31.82	2.35	12.99 (7574)	59.75	**2.64**	**1.09** (0)	34.85	2.05	**1.09** (0)	29.18	2.10
Road-Tx	4.28 (100)	49.65	**3.29**	**1.00** (0)	26.12	1.12	1.00 (0)	10.79	**0.82**	21.35 (141389)	83.42	**14.54**	1.03 (0)	32.61	**0.74**	1.03 (0)	6.11	**0.05**
Amazon0505	1080.62 (3425012)	818.36	**21.35**	39.89 (22483)	676.50	**1.86**	3.29 (0)	574.99	**1.62**	–	–	–	170.49 (249768)	–	**4.88**	4.19 (0)	–	**0.72**
Road-Asia	12.29 (2)	359.80	**7.89**	8.79 (0)	369.62	**6.12**	8.69 (0)	470.77	**7.04**	81.79 (98706)	406.40	**43.95**	8.69 (0)	401.60	**7.12**	8.79 (0)	415.44	**6.54**

Table 2: The running time of computing maximal k-plexes in real-world datasets

For each $k \in \{2, 3\}$, Table 3 indicates runtimes in seconds and the number of maximal exact k-plexes (in parenthesis) enumerated on each graph. Depending on the input graph and the value of the parameter α, it is not surprising that the set of maximal exact k-plexes is smaller than the whole set of maximal k-plexes. For $\alpha = 5$, it can be seen that the number of maximal exact 2-plexes is about 0.75% of the maximal 2-plexes for `Amazon0505`, while this number reached 67.43% for `Road-Pa` with $k = 3$.

4.3 Maximal Cohesive k-plexes Enumeration

In this subsection, we evaluate our approach for enumerating maximal cohesive k-plexes. Our comparative evaluation is made w.r.t. the number of (non)-cohesive k-plexes that are computed. Let us recall that maximal cohesive k-plexes are obtained by adding Constraint (7) to the SAT-based encoding of traditional maximal k-plexes. The comparison is done by setting k to 3 and α to 4. In fact, for $k <= 2$, all maximal k-plexes are clearly cohesive; and for $k = 3$ and for $5 \leq \alpha$, all maximal k-plexes are also cohesive. For a representative sample of datasets, Figure 2 provides comparative results w.r.t. the generated number of maximal (cohesive) k-plexes. It presents for each dataset the percentage of cohesive and non-cohesive among the set of all maximal k-plexes. As expected, all datasets contain non-cohesive maximal k-plexes. Especially for `Ca-Gr`, more than 65% are non-cohesive while this percentage is close to 20% for `Road-Pa` and `Road-Tx` datasets.

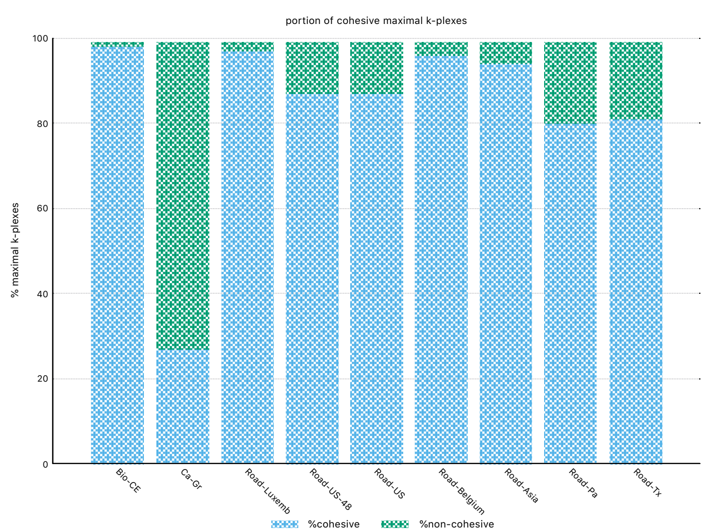

Figure 2: Cohesive vs non-cohesive maximal k-plexes

4.4 Maximal Cliques Enumeration

In this subsection, we present the empirical results to enumerate all maximal cliques on the different datasets fixing $k = 1$. We compare our SAT-based approach against the most recent algorithms for this task: D2K [10], EmMCE [7], NAUDE [27], and GP-B [35]. Table 4

Graph	2-plexes		3-plexes	
	$\alpha = 5$	$\alpha = 10$	$\alpha = 5$	$\alpha = 10$
Bio-CE-Gt	0.18 (65)	0.01 (0)	173.45 (11646)	0.01 (0)
Ca-Gr	0.08 (7)	0.00 (0)	18.38 (2697)	0.00 (0)
Bio-Dmela	0.89 (105)	0.01 (0)	1395.52 (544084)	0.01 (0)
Ca-Hep	0.00 (50)	0.01 (0)	73.66 (34635)	3.49 (0)
Gnutella	0.14 (0)	0.03 (0)	44.11 (71102)	0.01 (0)
As-Caida	209.88 (13603)	10.19 (279)	–	–
Road-Luxemb	0.06 (0)	0.03 (0)	0.29 (283)	0.00 (0)
Road-US-48	0.15 (0)	0.08 (0)	0.97 (3037)	0.01 (0)
Road-US	0.10 (0)	0.00 0	0.99 (3090)	0.01 (0)
Soc-Gemsec	4.43 (0)	0.62 0	954.69 (618521)	1.66 (0)
Amazon	6.88 (2320)	0.60 (0)	1592.90 (363866)	1.09 (0)
DBLP	53.74 (1254)	3.49 (0)	–	23.71 (0)
Road-Pa	2.88 (0)	0.00 (0)	12.18 (84737)	0.01 (0)
Road-Belgium	1.27 (0)	1.07 (0)	5.49 (7302)	0.01 (0)
Road-Tx	3.38 (0)	1.09 (0)	14.76 (95024)	1.09 (0)
Amazon0505	311.64 (25747)	21.03 (2)	–	30.36 (3)
Road-Asia	11.29 (0)	8.78 (0)	45.78 (96772)	8.88 (0)

Table 3: Results on maximal Exact k-plexes computation

Graph	SAPE	EmMCE	NAUDE	GP-B	D2K
Bio-CE	**0.01**	0.15	2.60	0.01	**0.01**
Bio-CE-Gt	**0.01**	0.12	0.06	0.04	**0.01**
Ca-Gr	0.04	2.72	10.32	0.09	**0.01**
Bio-Dmela	2.49	7.19	42.39	3.25	**0.03**
Ca-Hep	**0.01**	44.35	0.32	0.02	**0.01**
Gnutella	0.09	0.76	0.89	0.04	**0.03**
As-Caida	1.48	2.11	23.81	8.88	**0.08**
Road-Luxemb	0.28	14.81	138.71	0.85	**0.12**
Road-US48	0.38	2.05	76.07	**0.14**	0.17
Road-US	0.47	2.07	21.86	**0.15**	0.24
Soc-Gemsec	3.13	22.12	98.65	0.37	**0.09**
Amazon	2.93	44.44	102.78	18.27	**1.26**
DBLP	3.36	32.75	64.41	21.97	**1.31**
Road-Pa	**3.60**	95.92	154.81	47.29	3.80
Road-Belgium	4.59	37.10	204.69	12.21	**1.72**
Road-Tx	4.59	100.78	71.09	43.06	**3.28**
Amazon5050	10.58	254.87	491.02	12.58	**10.42**
Road-Asia	15.66	46.84	690.36	98.33	**24.38**
Avg Time	2.98	35.55	109.74	13.37	**2.34**

Table 4: Experimental results on maximal cliques computation

contains the comparative results. Clearly, it is interesting to observe that our method outperforms the three baseline algorithms EmMCE, NAUDE and GP-B by comfortable margins on all real graphs. The exception is on Road-US and Road-US-48 where GP-B is much faster than the other algorithms. In terms of average performance, our approach outperforms GP-B by 348.65%, EmMCE by 1092.95%, and NAUDE by 3582.55%. Moreover, our SAPE algorithm achieves competitive running time against D2K on various graphs (i.e., D2K slightly better than SAPE on average). For instance, SAPE spends less than 0.1 second to solve `Ca-Hep`, `Gnutella`, `Ca-Gr`, `Bio-CE-Gt` and `Bio-CE` datasets, and the obtained runtimes are very close to the ones by D2K algorithm. Overall, our SAPE approach is the second-fastest method next to D2K, at least 4 times faster on average than the third fastest method, GP-B, and about 11 times faster on average than the fourth fastest method, EmMCE, and 36 times faster on average than NAUDE.

5 RELATED WORK

Maximal k-plex enumeration. Many proposals for finding out all maximal k-plexes in graphs have been developed in the literature. A first method, introduced by [36], is based on the well-known Bron-Kerbosch algorithm [5]. Further, [8] proposed a framework for computing maximal subgraphs w.r.t. (connected) hereditary graph properties. Moreover, the authors of [3] proposed an algorithm, based on the method of [8], to compute the maximal (connected) k-plexes. In [35], Wang et al. proposed a parallel algorithm, based on a recursive decomposition of the original graph, to computing maximal cliques and k-plexes. Furthermore, in [9], the authors introduced a new algorithm for enumerating k-plex subgraphs larger than a fixed size. More recently, Zhou et al. [38] studied a novel algorithm for finding maximal k-plexes with predefined size. Note that the scalability issue is the main bottleneck of most of these state-of-the-art enumeration algorithms. In addition, all these proposals ignore the enumeration of cohesive k-plexes which frequently appear in real-world communities.

Maximal clique enumeration. The maximal clique enumeration problem has been the area of research of an extensive study in graph mining community. Several approaches are introduced to solve this problem. Fakhfakh et al. [13] gave a survey of the different work on this problem. Traditional algorithms for computing maximal cliques in graphs are based on various pruning techniques to decrease the search space and reduce the execution time [6]. In [35], Wang et al. proposed a novel algorithm based on iterative graph partitioning techniques to compute the set of cliques in real graphs. Unfortunately, most of these designed algorithms suffer from a degradation in scalability for large scale graphs.

6 CONCLUSION AND FUTURE WORK

This paper presented the first declarative SAT-based approach for enumerating all maximal k-plexes in large graphs as well as the novel proposed structure called cohesive k-plexes. The problem is modeled as a propositional formula, whose models are the maximal (cohesive) k-plexes of interest. Then, to exhibit the nice declarative features of our framework, we showed how the particular cases of maximal cliques and maximal exact k-plexes can be found with a slight modification of the initial SAT-based encoding. Last, to enhance our polynomial SAT-based encoding, we harnessed a decomposition technique, leading to a highly competitive approach w.r.t. the state-of-the-art algorithms.

We identified several directions for future work. We plan to propose a parallelization approach to improve the efficiency of our proposed SAT-based framework. We also would like to extend our proposal to enumerate *maximum* k-plexes in large scale graphs.

REFERENCES

[1] Olivier Bailleux and Yacine Boufkhad. 2003. Efficient CNF Encoding of Boolean Cardinality Constraints. In *CP*. 108–122.

[2] Balabhaskar Balasundaram, Sergiy Butenko, and Illya V. Hicks. 2011. Clique Relaxations in Social Network Analysis: The Maximum *k*-Plex Problem. *Operations Research* 59, 1 (2011), 133–142.

[3] Devora Berlowitz, Sara Cohen, and Benny Kimelfeld. 2015. Efficient Enumeration of Maximal k-Plexes. In *SIGMOD*. 431–444.

[4] Abdelhamid Boudane, Saïd Jabbour, Badran Raddaoui, and Lakhdar Sais. 2018. Efficient SAT-Based Encodings of Conditional Cardinality Constraints. In *LPAR*. 181–195.

[5] Coen Bron and Joep Kerbosch. 1973. Algorithm 457: Finding All Cliques of an Undirected Graph. *Commun. ACM* 16, 9 (Sept. 1973), 575–577.

[6] Frédéric Cazals and Chinmay Karande. 2008. A note on the problem of reporting maximal cliques. *Theor. Comput. Sci.* 407, 1-3 (2008), 564–568.

[7] James Cheng, Yiping Ke, Ada Wai-Chee Fu, Jeffrey Xu Yu, and Linhong Zhu. 2011. Finding Maximal Cliques in Massive Networks. *ACM Trans. Database Syst.* 36, 4 (2011).

[8] Sara Cohen, Benny Kimelfeld, and Yehoshua Sagiv. 2008. Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties. *J. Comput. Syst. Sci.* 74, 7 (2008), 1147–1159.

[9] Alessio Conte, Donatella Firmani, Caterina Mordente, Maurizio Patrignani, and Riccardo Torlone. 2017. Fast Enumeration of Large k-Plexes. In *KDD*. 115–124.

[10] Alessio Conte, Tiziano De Matteis, Daniele De Sensi, Roberto Grossi, Andrea Marino, and Luca Versari. 2018. D2K: Scalable Community Detection in Massive Networks via Small-Diameter k-Plexes. In *KDD*. 1272–1281.

[11] Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. 2014. Local search of communities in large graphs. In *SIGMOD*. 991–1002.

[12] Niklas Eén and Niklas Sörensson. 2006. Translating Pseudo-Boolean Constraints into SAT. *JSAT* 2, 1-4 (2006), 1–26.

[13] Faten Fakhfakh, Mohamed Tounsi, Mohamed Mosbah, and Ahmed Hadj Kacem. 2018. Algorithms for Finding Maximal and Maximum Cliques: A Survey. 745–754.

[14] Tias Guns, Siegfried Nijssen, and Luc De Raedt. 2011. Itemset mining: A constraint programming perspective. *Artif. Intell.* 175, 12-13 (2011), 1951–1983.

[15] R. Gupta and J. Walrand. 2004. Approximating maximal cliques in ad-hoc networks. In *ISPIM*, Vol. 1. 365–369.

[16] Yacine Izza, Saïd Jabbour, Badran Raddaoui, and Abdelhamid Boudane. 2020. On the Enumeration of Association Rules: A Decomposition-based Approach. In *IJCAI*. 1265–1271.

[17] Saïd Jabbour, Jerry Lonlac, Lakhdar Sais, and Yakoub Salhi. 2014. Extending modern SAT solvers for models enumeration. In *IEEE International Conference on Information Reuse and Integration*. 803–810.

[18] Saïd Jabbour, Nizar Mhadhbi, Badran Raddaoui, and Lakhdar Sais. 2020. SAT-based models for overlapping community detection in networks. *Computing* 102, 5 (2020), 1275–1299.

[19] Saïd Jabbour, Lakhdar Saïs, and Yakoub Salhi. 2014. A Pigeon-Hole Based Encoding of Cardinality Constraints. In *ISAIM*.

[20] Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. 2017. Mining Top-k motifs with a SAT-based framework. *Artif. Intell.* 244 (2017), 30–47.

[21] Hua Jiang, Chu-Min Li, Yanli Liu, and Felip Manyà. 2018. A Two-Stage MaxSAT Reasoning Approach for the Maximum Weight Clique Problem. In *AAAI*. 1338–1346.

[22] Tuukka Korhonen, Jeremias Berg, and Matti Järvisalo. 2019. Enumerating Potential Maximal Cliques via SAT and ASP. In *IJCAI*. 1116–1122.

[23] Jussi M. Kumpula, Mikko Kivela, Kimmo Kaski, and Jari Saramaki. 2008. Sequential algorithm for fast clique percolation. *Physical Review* E 78 (2008). Issue 2.

[24] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[25] Chu Min Li and Zhe Quan. 2010. An Efficient Branch-and-Bound Algorithm Based on MaxSAT for the Maximum Clique Problem. In *AAAI*.

[26] Zhuqi Miao and Balabhaskar Balasundaram. 2017. Approaches for finding cohesive subgroups in large-scale social networks via maximum *k*-plex detection. *Networks* 69, 4 (2017), 388–407.

[27] Naudé and A. Kevin. 2016. Refined Pivot Selection for Maximal Clique Enumeration in Graphs. *Theor. Comput. Sci.* 613 (2016), 28–37.

[28] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *nature* 435 (2005), 814–818.

[29] David A. Plaisted and Steven Greenbaum. 1986. A Structure-preserving Clause Form Translation. *J. Symb. Comput.* 2, 3 (Sept. 1986), 293–304.

[30] Arnau Prat-Pérez, David Dominguez-Sal, and Josep-Lluís Larriba-Pey. 2014. High quality, scalable and parallel community detection for large real graphs. In *WWW*. 225–236.

[31] Oleg Rokhlenko, Ydo Wexler, and Zohar Yakhini. 2007. Similarities and differences of gene expression in yeast stress conditions. *Bioinformatics* 23, 2 (2007), 184–190.

[32] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*.

[33] Stephen Seidman and Brian Foster. 1978. A graph-theoretic generalization of the clique concept*. *Journal of Mathematical Sociology* 6 (01 1978), 139–154.

[34] Carsten Sinz. 2005. Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In *CP*. 827–831.

[35] Zhuo Wang, Qun Chen, Boyi Hou, Bo Suo, Zhanhuai Li, Wei Pan, and Zachary G. Ives. 2017. Parallelizing maximal clique and k-plex enumeration over graph data. *J. Parallel Distrib. Comput.* 106 (2017), 79–91.

[36] Bin Wu and Xin Pei. 2007. A Parallel Algorithm for Enumerating All the Maximal k-Plexes. 476–483.

[37] Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowl. Inf. Syst.* 42, 1 (2015), 181–213.

[38] Yi Zhou, Jingwei Xu, Zhenyu Guo, Mingyu Xiao, and Yan Jin. [n.d.]. Enumerating Maximal *k*-Plexes with Worst-Case Time Guarantee. In *AAAI*. 2442–2449.