On Maximal Cliques with Connectivity Constraints in Directed Graphs*

Alessio Conte[1], Mamadou Moustapha Kanté†[2], Takeaki Uno[3], and Kunihiro Wasa[4]

1 **Università di Pisa, Pisa, Italy**
conte@di.unipi.it
2 **Université Clermont Auvergne, LIMOS, CNRS, Aubière, France**
mamadou.kante@uca.fr
3 **National Institute of Informatics, Tokyo, Japan**
uno@nii.ac.jp
4 **National Institute of Informatics, Tokyo, Japan**
wasa@nii.ac.jp

Abstract

Finding communities in the form of cohesive subgraphs is a fundamental problem in network analysis. In domains that model networks as undirected graphs, communities are generally associated with dense subgraphs, and many community models have been proposed. Maximal cliques are arguably the most widely studied among such models, with early works dating back to the '60s, and a continuous stream of research up to the present. In domains that model networks as *directed* graphs, several approaches for community detection have been proposed, but there seems to be no clear model of cohesive subgraph, *i.e.*, of what a community should look like. We extend the fundamental model of clique to directed graphs, adding the natural constraint of strong connectivity within the clique. We characterize the problem by giving a tight bound for the number of such cliques in a graph, and highlighting useful structural properties. We then exploit these properties to produce the first algorithm with polynomial delay for enumerating maximal strongly connected cliques.

1998 ACM Subject Classification G.2.2 Graph Theory, Graph algorithms

Keywords and phrases Enumeration algorithms, Bounded delay, Directed graphs, Community structure, Network analytics

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2017.23

1 Introduction

The problem of community detection in graphs has been extensively studied. In undirected graphs, dense subgraphs are often used to detect communities, with applications in areas such as social network analysis [25, 28], biology [16], and more [13].

Several definitions of dense subgraph have been proposed to model communities [22, 28]. The earliest, and perhaps the most widely studied is that of the maximal clique: interest in the problem of finding maximal cliques started several decades ago [1, 5, 21] and effort to produce efficient algorithms can still be seen in recent works [8, 10, 12].

* This work was supported by JST CREST, Grant Number JPMJCR1401, Japan.
† M.M. Kanté is supported by French Agency for Research under the GraphEN project (ANR-15-CE-0009).

As for directed graphs, there seems to be consensus in literature [18, 19, 23] on the fact that ignoring edge directions and applying community detection techniques for undirected graphs is not satisfactory. Several ad-hoc techniques for clustering and community discovery have been proposed, mirroring the goals of algorithms for undirected graphs.

Awerbuch *et al.* [3] proposed a bounded-error scheme for aggregating vertices in directed graphs as a hierarchical structure. Leicht *et al.* [19] adapted the concept of *modularity* to account for edge directions, with the aim of extracting more meaningful clusters. The LinkRank algorithm [18] aimed at partitioning a directed graph into communities using random walks and the PageRank algorithm. More approaches can be found in [13].

Subgraph-based community models in undirected networks are thoroughly studied in community detection (and network analysis in general), thus it would be natural to imagine that similar models were object of study in the directed area. Surprisingly, this seems to be a road less traveled.[1] Charikar *et al.* [6] considered communities in directed graphs as sets of vertices whose induced subgraphs have many edges, regardless of connectivity. The well-known work by Kleinberg *et al.* [17] defined a community in the web graph with respect to a topic as a special *bipartite clique* $K_{i,j}$, in which each of the i vertices has edges *towards* each of the other j vertices, which represent authority pages on the topic. To the best of our knowledge, there are no other community models for directed graphs that are widely accepted and rigorous. This motivates our interest in combining the basic maximal clique model with connectivity in directed graphs, that is strong connectivity. We call this model a *strongly connected clique* (SCQ for short), and investigate both its properties and the problem of efficiently finding all maximal SCQs.

Generic enumeration techniques for maximal subgraphs have been proposed for *strongly accessible* properties [2], *i.e.*, such that every non-maximal subgraph A which verifies the property is included in a subgraph B of size exactly $|A| + 1$ that also verifies the property. Cohen *et al.* [7] proposed an algorithmic framework for enumerating maximal subgraphs with respect to subsets of strongly accessible properties, namely *hereditary* and *connected-hereditary* graph properties. SCQs, however, fit in neither of these classes.

Finding maximal subgraphs satisfying a non *accessible* property is a challenging task, as their structure is unsystematic, and their enumeration requires new techniques and theoretical insight. In this work, we show that SCQs have a peculiar but rigorous structure, which fits under a relaxed, more general notion of accessibility. We then exploit this structure to design SCQ-ENUM, an efficient algorithm that enumerates SCQs with *delay* bounded by $O(\min(\omega(G)d^2\Delta^2, m^2))$, where $\omega(G)$, d, Δ and m are respectively the largest size of an SCQ, degeneracy, maximum degree and number of edges of the input graph, and the delay is the maximum time elapsed between two consecutive outputs. The value of SCQ-ENUM is two-fold: on one hand, it constitutes a first step towards the characterization, and potentially towards general enumeration techniques, for a wider range of problems that are not accessible. On the other hand, SCQ-ENUM is also an efficient practical tool for discovering community structures in directed networks. Finally, we complete the analysis of the model by giving a tight bound for the number of maximal SCQs in an n-vertex graph.

[1] It should be mentioned that strongly connected components have been object of thorough study, however these may be very large, sparse, and thus may not be significant indicators of community structures.

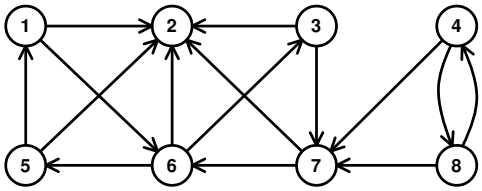

Figure 1 A graph with 4 maximal SCQs ({1,5,6},{2},{3,6,7},{4,8}) whose underlying undirected graph has 3 maximal cliques ({1,2,5,6},{2,3,6,7},{4,7,8}).

2 Preliminaries

We refer to [11] for our graph terminology, and all (directed) graphs considered in this paper are without multi-edges and loops (but may contain edges of opposite direction). The vertex set of a graph G is denoted by $V(G)$ and its edge set by $E(G)$. A (directed) edge (or arc) from x to y is denoted by (x, y) in which x is the *tail* and y the *head*; we will say that the edge is *from* x and *towards* y. When $E(G)$ is symmetric, *i.e.*, $(x, y) \in E(G)$ if and only if $(y, x) \in E(G)$, we call G *undirected* and denote each edge (x, y) of G by xy (equivalently yx). For a graph G, we denote by $\mathsf{u}(G)$, called *underlying (undirected) graph* of G, the undirected graph with vertex set $V(G)$ and edge set $\{xy \mid (x, y) \in E(G) \text{ or } (y, x) \in E(G)\}$. We use n and m to denote the number of vertices and edges, respectively, in any graph.

For a vertex x in a graph G, $N_G(x)$ denotes its set of neighbours, which includes both *in-neighbours*, *i.e.*, $\{y \in V(G) \mid (y, x) \in E(G)\}$, and *out-neighbours*, *i.e.*, $\{y \in V(G) \mid (x, y) \in E(G)\}$. $|N_G(x)|$ denotes the *degree* of x, and $\Delta(G)$ the maximum degree of a vertex in G. Any vertex with degree $|V(G)| - 1$ is called a *universal* vertex. The subgraph of G induced by $X \subseteq V(G)$ is the graph $G[X] = (X, E(G) \cap (X \times X))$. When the graph G is clear from the context we will drop the subscripts from the notations $N_G(x)$ and similar ones, and also write V (or similar notations E, Δ, ...) instead of $V(G)$ (or $E(G)$, $\Delta(G)$, ...).

The power-set of the set V is denoted by 2^V. For two sets of vertices A and B we denote by $A \setminus B$ the set $\{x \in A \mid x \notin B\}$. Given a total ordering on the vertices in V, represented by increasing labels $v_1, \ldots, v_n$, the associated *lexicographic ordering* on 2^V, denoted by $\leq$, is such that $A \leq B$ if A contains the smallest element not in common, *i.e.*, $\min((A \cup B) \setminus (B \cap A)) \in A$.

A *clique* of an undirected graph G is a subset C of $V(G)$ that induces a complete graph, and a *maximal clique* is a clique C of G such that $C \cup \{x\}$ is not a clique for all $x \in V(G) \setminus C$. Let $G = (V, E)$ be a (directed) graph. A *strongly connected clique* (or SCQ for short) is a set $C \subseteq V(G)$ such that $G[C]$ is strongly connected, and $\mathsf{u}(G)[C]$ is a clique of $\mathsf{u}(G)$. We recall that a directed graph (or subgraph) is strongly connected if and only if for each bipartition (V_1, V_2) of V there is an edge from V_2 to V_1 (and symmetrically from V_1 to V_2) [9]. We assume that a single vertex is an SCQ; we denote the maximum size of an SCQ in G by $\omega(G)$, and the maximum size of a clique in $\mathsf{u}(G)$ by $\omega(\mathsf{u}(G))$. It is worth noticing that if G is undirected, SCQs and cliques coincide. An SCQ C is maximal if there is no SCQ C' such that $C \subset C'$. Given $C \subseteq V$, the set $X \subseteq V \setminus C$ is *addible* to C if $C \cup X$ is an SCQ, and $Y \subseteq C$ is *removable* from C if $C \setminus Y$ is an SCQ. Furthermore, we say that a vertex x is a *sink w.r.t.* C if there is no $(x, y) \in E(G)$ with $y \in C$, and a *source w.r.t.* C if there is no $(y, x) \in E(G)$ with $y \in C$. A graph with its maximal SCQs and cliques is shown in Figure 1.

A graph G is *d-degenerate* if each induced subgraph of G has a vertex of degree at most d. The *degeneracy* of a graph G is the minimum d such that G is d-degenerate. A *degeneracy ordering* of a graph G of degeneracy d is a sequence $v_1, v_2, \ldots, v_n$ of its vertex set such

that the degree of each v_i in $G[\{v_i, \ldots, v_n\}]$ is at most d; we call $N(v_i) \cap \{v_{i+1}, \ldots, v_n\}$ the *forward neighbours* of v_i. We assume that any graph is given with a degeneracy ordering[2].

3 Problem Characterization

Maximal SCQs are a challenging problem as they do not satisfy the *strong accessibility* property. However, we provide some related properties that will be the key of our enumeration algorithm.

3.1 Relaxed Accessibility of Strongly Connected Cliques

We recall that a *set system* $(V, \mathcal{E} \subseteq 2^V)$ is *(weakly) accessible* if for each $X \in \mathcal{E}$, there is $x \in X$ such that $X \setminus \{x\} \in \mathcal{E}$, and it is *strongly accessible* if in addition for each $X, Y \in \mathcal{E}$ with $Y \subset X$, there is $x \in X \setminus Y$ such that $Y \cup \{x\} \in \mathcal{E}$. In both cases it is assumed that $\emptyset \in \mathcal{E}$. The following two lemmas prove a relaxed notion of weak and strong accessibility.

▶ **Lemma 1.** *Let C be a non-empty* SCQ *of G. There exists $Z \subseteq C$ removable from C and such that $|Z| \leq 2$.*

Proof. As a single vertex and the empty set are SCQs, if $|C| = 3$ any $Z \subset C$ with $|Z| = 2$ is removable, and if $|C| = 1$ or $|C| = 2$ any vertex in C is removable. Suppose then that $|C| \geq 4$ and let us prove that it has a removable vertex.

Let y be an arbitrary vertex of C and suppose that $C' = C \setminus y$ is not strongly connected. Then, there exists a bipartition (X, Y) of C' such that $E(G[C]) \cap (Y \times X) = \emptyset$. As $C = C' \cup \{y\}$ is strongly connected, we must have $w \in X$ and $w' \in Y$ such that $(y, w), (w', y) \in E(G[C])$, and y can reach every vertex in X, and also every vertex from Y can reach y. Since, $|C| \geq 4$ and thus $|C'| \geq 3$, either $|X| \geq 2$ or $|Y| \geq 2$. Assume $|X| \geq 2$: let $z \in X$ be a leaf of a traversal of $X \cup \{y\}$ starting from y (recall that y can reach all vertices in X). As z is a leaf, if we remove it, y can still reach all vertices in $X \setminus \{z\}$. Furthermore, each vertex in $X \setminus \{z\}$ has an edge towards *every* vertex in Y, as C is an SCQ, and every vertex in Y can reach y. Thus $\{y\} \cup (X \setminus \{z\}) \cup Y = C \setminus \{z\}$ is an SCQ, *i.e.*, $Z = \{z\}$ is a removable set in C with $|Z| \leq 2$. If $|X| = 1$, then $|Y| \geq 2$, and the proof is symmetrical by choosing z as a leaf vertex in a traversal of $G[Y \cup \{y\}]$ with the edges reversed, starting from y. ◀

▶ **Lemma 2.** *Given two* SCQs *C and D such that $D \subset C$, there exists $X \subseteq C \setminus D$ addible to D with $|X| \leq 2$.*

Proof. If $|C \setminus D| \leq 2$ the lemma is trivially true, so assume $|C \setminus D| \geq 3$. Any vertex in $C \setminus D$ with edges both towards and from vertices in D is addible to D, so assume that no such vertex exists. Hence, all the vertices in $C \setminus D$ are either sinks or sources *w.r.t.* D, that we denote by K and R, respectively. Any set $\{k, r\}$ with $k \in K$, $r \in R$ and $(k, r) \in E(G[C])$ is an addible set to D. Assume then that no such set does exist: all the vertices in K cannot reach R or D, and all the vertices in R cannot be reached from K or D. This is a contradiction as $C = D \cup K \cup R$ is strongly connected, thus an addible set $\{k, r\}$ exists. ◀

And as any non-maximal SCQ is contained in a larger one, we obtain the following.

▶ **Corollary 3.** *An* SCQ *C is maximal in G if and only if there is no $X \subseteq V(G) \setminus C$ addible to C with $|X| \leq 2$.*

[2] Such an ordering can be computed in linear time by iteratively removing the smallest degree vertex.

Thanks to Lemmas 1 and 2, we can say that the property of being an SCQ belongs to a relaxed class of accessibility, since (i) for each $X \in \mathcal{E}$, there is $Z \subseteq X$ such that $X \setminus Z \in \mathcal{E}$, and (ii) for each $X, Y \in \mathcal{E}$ with $Y \subseteq X$ there is $Z \in X \setminus Y$ such that $Y \cup Z \in \mathcal{E}$, where the size of Z is at most 2. This is a generalization of the definitions of strong and weakly accessible classes, which are obtained from the above by simply setting $|Z| = 1$.

3.2 Maximal Undirected and Strongly Connected Cliques

On top of the accessibility of the problem, we are interested in studying the relationship between the SCQs in G and the cliques in the underlying undirected graph $\mathsf{u}(G)$. Lemma 4 highlights the first basic, but important, relationship.

▶ **Lemma 4.** *Given a directed (not necessarily strongly connected) clique D, the strongly connected components of D are the maximal* SCQ*s in D.*

Proof. Any SCQ C is contained in a strongly connected component of a directed clique by definition, as C is strongly connected and $\mathsf{u}(G[C])$ is a clique. Furthermore, an SCQ may not contain vertices from different strongly connected components, as it would not be strongly connected. Thus, a strongly connected component of D is a maximal SCQ in D. ◀

▶ **Corollary 5.** *A directed clique D contains at most $|D|$ maximal* SCQ*s, which are disjoint.*

3.3 Bounding the Number of Maximal SCQs

For a graph G, let us denote by $gc(G)$ and $gc(\mathsf{u}(G))$ the number of maximal SCQs in G and maximal cliques in $\mathsf{u}(G)$ respectively, and for n, let us denote by $g(n)$ the maximum number of maximal SCQs in an n-vertex graph. From Corollary 5, any maximal SCQ in a directed graph G is contained in a maximal clique of $\mathsf{u}(G)$, *i.e.*, $gc(G) \leq \omega(\mathsf{u}(G)) \cdot gc(\mathsf{u}(G))$. But the number of SCQs in a graph can be much smaller than the number of cliques of its underlying undirected graph: For instance, an n-vertex DAG has exactly n maximal SCQs of size 1 while the number of maximal cliques of its underlying undirected graph can be arbitrarily large. As the maximum number of maximal cliques in an undirected n-vertex graph is $3^{\frac{n}{3}}$ [21], we can immediately conclude that $3^{\frac{n}{3}} \leq g(n) \leq n \times 3^{\frac{n}{3}}$. We can adapt the proof from [21] to show that $g(n)$ is indeed $3^{\frac{n}{3}}$.

Let G be an n-vertex graph, and x, y two vertices of G, and let $G(x; y)$ be defined similarly to [21], as the graph obtained by removing all edges incident to x, and replacing them so that the neighbourhood of x is identical to that of y, *i.e.*, $(x, v) \in E(G(x; y))$ iff $(y, v) \in E(G)$ and $(v, x) \in E(G(x; y))$ iff $(v, y) \in E(G)$. Let $\chi(x)$ be the number of maximal SCQs containing x, let $\alpha(x)$ be the number of new maximal SCQs created by removing x (*i.e.*, subsets of SCQs containing x which become now maximal), and $\beta(x)$ the number of SCQs which are not maximal anymore after removing x[3]. It is straightforward to see that if x and y are not adjacent, the number of SCQs in $G(x; y)$ is given by $gc(G) + \chi(y) - \chi(x) + \alpha(x)$. Indeed all SCQs containing x have been removed, and replaced by $\alpha(x)$ new maximal cliques; furthermore, for each of the $\chi(y)$ maximal SCQs containing y in G, we now have a new one containing x instead of y. If x and y are adjacent, any maximal SCQ containing y in $G(x; y)$ will be simply incremented with x; as a result, the number of maximal SCQs in $G(x; y)$ will

[3] Note that in the undirected case $\alpha(x)$ is bounded by $\chi(x)$, but in the directed case $\alpha(x)$ may be larger than $\chi(x)$ by up to an n factor.

be only $gc(G) - \chi(x) + \alpha(x)$. We are now ready to characterize the graph with the highest number of maximal SCQs.

▶ **Lemma 6.** *Let G be a graph on $n > 4$ vertices with $gc(G) = g(n)$. There exists G^*, an n-vertex graph such that $gc(G^*) = g(n)$ and $\mathsf{u}(G^*)$ is a* complete multipartite graph *with no universal vertices.*

Proof. If $\mathsf{u}(G)$ is a clique, it has at most n maximal SCQs by Lemma 4, so we can replace G with any DAG and still have n maximal SCQs. Thus we can assume that $\mathsf{u}(G)$ is not a clique, and has at least 2 non-universal vertices (*i.e.*, not connected to every other vertex).

For two non-adjacent vertices x and y, we know that $G(x; y)$ and $G(y; x)$ cannot have more SCQs than G. As $\alpha(x) \geq 0$, we have $\chi(x) = \chi(y)$ for any pair of non-adjacent vertices. This implies $\alpha(x) = 0$ for every non-universal vertex x. Thus, if x and y are non-adjacent, $gc(G(x; y)) = gc(G) = g(n)$. From G, we can obtain the graph G^* as follows: for each vertex x, and for each vertex y non-adjacent to x, iteratively replace G with $G(y; x)$.

Observe from the discussion above that $gc(G^*) = gc(G) = g(n)$. Also, $\mathsf{u}(G^*)$ is a complete multipartite graph. Indeed, as each pair of non-adjacent vertices has the same neighbours, we can partition the graph into independent sets such that two vertices in two different independent sets are adjacent. Again, if $\mathsf{u}(G^*)$ is a clique (this may be the case for $n = 4$), we replace G^* with any DAG without compromising the number of maximal SCQs and thus obtaining at least 2 non-universal vertices.

Assume G^* has a universal vertex v. Removing v can decrease the number of maximal SCQs by at most 1. In fact, any SCQ C that is non maximal after removing v, is included in a larger SCQ C', to which v can be added as it is in the same strongly connected component as C (which is a subset of C') and it is connected to all vertices of C'. Thus the only maximal SCQ that can may be lost is the one made by only v, if it is a maximal SCQ.

Let A be the set of universal vertices of G^*. If $|A| > 1$ we can simply remove the edges between two vertices of A and replace each edge (a, b), for $a \in A$ and $b \in V(G^*) \setminus A$, by (b, a). If $|A| = 1$, *i.e.*, $A = \{v\}$ for some v, let us take any independent set I in G^* of size at least 2 (which exists because v is the only universal vertex), and then remove all the edges between v and vertices in I, and replace any edge (v, b), for $b \in V(G^*) \setminus I \cup \{v\}$, by (b, v). As a sink is a maximal SCQ, we can conclude from the paragraph above that the number of SCQs of the obtained graph is still equal to $gc(G) = g(n)$. Since, the underlying undirected graph of this obtained graph is a complete multipartite graph with no universal vertices, we are done. ◀

Thanks to Lemma 6, we can link the number of maximal SCQs in G^* to the number of maximal cliques in $\mathsf{u}(G^*)$. This relation will enable us to give a tight bound for $g(n)$.

▶ **Lemma 7.** *Let $G = (V, E)$ be a graph such that $\mathsf{u}(G)$ is a complete multipartite graph with no universal vertices. Then $gc(G) \leq gc(\mathsf{u}(G))$.*

Proof. Let $\mathcal{S} = \{S_1 \dots S_k\}$ be the set of maximal independent sets of $\mathsf{u}(G)$, which forms a k-partition of V. Let $s(v)$ be the size of the unique maximal independent set S_i containing v; as G and $\mathsf{u}(G)$ have no universal vertices, $s(v) \geq 2$. By definition each maximal SCQ in G is a subset of some maximal clique of $\mathsf{u}(G)$, and recall that each maximal clique of $\mathsf{u}(G)$ (a complete multipartite graph) is obtained by selecting exactly one vertex from each of its maximal independent sets.

Let the occurrence $mc(C)$ of an SCQ C be the number of maximal cliques of $\mathsf{u}(G)$ that contain C, and let the *weight* $w(C)$ of C be $\frac{1}{mc(C)}$, or 0 if $C = \emptyset$. For a maximal clique Q of $\mathsf{u}(G)$, let the weight $w(Q)$ of Q be instead $\sum_{X \text{ a maximal SCQ of } G[Q]} w(X)$. The sum of the weights of all maximal cliques in $\mathsf{u}(G)$ will be *at least* equal to the number of maximal SCQs in

G: any maximal SCQ C will be considered $mc(C)$ times, each time adding $w(C) = 1/mc(C)$, for a total contribution of 1. This sum may be larger, as it can include subsets of maximal cliques which are not maximal SCQs in G, but cannot be smaller.

Let C be a maximal SCQ and let $\mathcal{T} \subseteq \mathcal{S}$ be the maximal independent sets that do not contain any vertex of C. Then, the maximal cliques that contain C are all the ones obtained by adding a single vertex from each independent set in $\mathcal{T}$, thus $mc(C) = \prod_{S_i \in \mathcal{T}} |S_i|$. This means that adding a vertex v to C reduces $mc(C)$ and increases $w(C)$ by a factor $s(v)$.

Let us now consider the highest possible weight of a maximal clique Q in $\mathsf{u}(G)$. Note that, by Corollary 5, the maximal SCQs within Q are at most $|Q|$ and do not overlap. If Q contains a single maximal SCQ X, we have $|X| = |Q|$, $mc(X) = 1$ and $w(X) = 1$, thus $w(Q) = 1$. Otherwise, let X and Y be two maximal SCQs in Q, X being the one with highest weight. Note that $w(X) + w(Y) \leq 2w(X)$. Assume that we could remove a vertex v from Y and add it to X, obtaining X' and Y': we have $w(X') = s(v) \cdot w(X)$, and as $s(v) \geq 2$, $w(X') + w(Y') = w(X) \cdot s(v) + w(Y') \geq 2w(X) \geq w(X) + w(Y)$. This hypothetical operation can increase the total weight of Q but not decrease it, *i.e.*, for any distribution of maximal SCQs in Q, a different that has the size of the largest SCQ increased by one, and that of another one reduced by one, has greater or equal weight. We can repeat this hypothetical step, iteratively enlarging X until we will finally consider a distribution with a single maximal SCQ X of size $|X| = |Q|$. As $w(Q)$ in this case is at least as large as that obtained with any other distribution of maximal SCQs in Q, and as shown above $w(Q) = 1$ in this case, we have $w(Q)$ is always at most 1. Therefore, the number of maximal SCQs in G, *i.e.*, the sum of all weights of the maximal cliques in $\mathsf{u}(G)$ cannot be larger than $gc(\mathsf{u}(G))$. ◀

It is known that the undirected graph with the highest number of maximal cliques is the Moon-Moser graph [21], which is a complete multipartite graph in which as many maximal independent sets as possible have size 3, while the remaining ones may only have size 2.[4] Clearly, such a graph does not have universal vertices, and thus is compatible with the definition of G^*. We can thus say that the underlying graph $\mathsf{u}(G)$ of the graph G^* with the highest number of maximal SCQs will be a Moon-Moser graph. Furthermore, by Lemma 7 we get the upper bound $g(n) = gc(G^*) \leq gc(\mathsf{u}(G^*))$.

It is now easy to prove that this is a tight bound: when G is symmetric (*i.e.*, $(x, y) \in E(G)$ iff $(y, x) \in E(G)$) then the connectivity in G is the same as in $\mathsf{u}(G)$ and each maximal clique in $\mathsf{u}(G)$ will be a maximal SCQ in G. Thus we have $g(n) = gc(G^*) = gc(\mathsf{u}(G^*))$. By combining this lower bound with Lemma 7 and the Moon-Moser bound [21], we can conclude the following (the case $2 \leq n \leq 4$, not covered by Lemma 6, is omitted for space reasons, but can be trivially verified).

▶ **Theorem 8.** *For every integer $n > 1$,*

$$g(n) = \begin{cases} 3^{\frac{n}{3}} & \text{if } n \equiv 0 \pmod 3, \\ \frac{4}{3} \cdot 3^{\lfloor \frac{n}{3} \rfloor} & \text{if } n \equiv 1 \pmod 3 \\ 2 \cdot 3^{\lfloor \frac{n}{3} \rfloor} & \text{if } n \equiv 2 \pmod 3. \end{cases}$$

Finally, the same result can be proven for *oriented* graphs, that are directed graphs where each edge may only have one direction, as long as n is not 5 or 6 (this is omitted for space reasons but also involves suitable orientations of Moon-Moser graphs).

[4] Equivalently, the remaining ones may have size 4. However it is not necessary to consider this case.

4 Listing Maximal SCQs

While the number of maximal SCQs in a graph G is at most n times the number of maximal cliques of its underlying (undirected) graph $\mathsf{u}(G)$, and each maximal SCQ of G is contained in some maximal clique of $\mathsf{u}(G)$, one *cannot* efficiently use output-polynomial algorithms for listing maximal cliques in undirected graphs in order to list the maximal SCQs of a graph. For example, any orientation of a Moon-Moser n-vertex graph into a DAG has exactly n maximal SCQs, while its underlying graph has $\Theta(3^{\frac{n}{3}})$ maximal cliques. This strategy would hence take exponential running time to find just a linear number of maximal SCQs.

In this section we design an algorithm that enumerates all maximal SCQs of a graph $G = (V, E)$ with polynomial delay.

Intuitively, given a maximal SCQ (called sometimes a solution) S, our algorithm uses the vertices in $V \setminus S$ to find other solutions similar to S; we refer to this process as *visiting* S. By visiting these newly found solutions the algorithm eventually finds all solutions in G.

4.1 Algorithm Description

The algorithm, which we call SCQ-ENUM, is described in Algorithm 1. SCQ-ENUM uses a RESULT set, which will store all solutions found so far. The primitive $\mathit{contains}(S, \text{RESULT})$ is a subroutine that returns true if $S \in$ RESULT, *i.e.*, S has already been found and does not need to be visited again, and the primitive $\mathit{add}(S, \text{RESULT})$ adds S to the RESULT set. Finally, SCQ-ENUM exploits the function $\text{COMPLETE}(X, A)$, which will iteratively add the lexicographically minimum addible vertex or pair of vertices from A to a SCQ X, until X is maximal *w.r.t.* A, and return it. For brevity, $\text{COMPLETE}(X)$ represents $\text{COMPLETE}(X, V)$. Thanks to the accessibility proven in Lemma 2 and Corollary 3, $\text{COMPLETE}(X)$ will surely return a maximal SCQ. We recall that we assume the graph given with the degeneracy ordering, and we consider that ordering and its associated lexicographic one in the algorithm (see Section 2). The primitive $\text{MIN-LEX}(\mathcal{T})$ finds the minimum in the collection $\mathcal{T} \subseteq 2^V$.

SCQ-ENUM is in the same spirit as the one for listing the maximal cliques in an undirected graph [10]. It does a DFS traversal of the graph of solutions where (S, S') is an edge if S' can be obtained from S by adding a new vertex (or a pair of vertices), removing its (their) non-neighbours and finally completing the obtained set into a maximal SCQ. Let us describe the algorithm.

SCQ-ENUM consists in calling the function $\text{ENUM}(S)$, with S a maximal SCQ. In turn, $\text{ENUM}(S)$ will find *all* solutions that have a non-empty overlap with S. The function will consider all vertices $x \in V \setminus S$, and for each of them will try to generate a new maximal SCQ containing x and some vertices of S: by calling $X \leftarrow \text{COMPLETE}(\{x\}, I)$ the algorithm will get the SCQ containing x, maximal *w.r.t.* the induced subgraph $G[S \cup \{x\}]$; note that there is only one such SCQ, as $G[S \cup \{x\}]$ is a clique in $\mathsf{u}(G[S \cup \{x\}])$ (see Lemma 4). Then X is extended with $\text{COMPLETE}(X)$ so that it is maximal *w.r.t.* G, *i.e.*, a solution. Then, in the second for loop, the same process is repeated for pairs of vertices rather than single vertices. Every time a solution S' is found, we recur in $\text{ENUM}(S')$, which will visit S', adding it to the RESULT set and finding more solutions starting from S'. If S' is already in the RESULT set, however, it means it was already visited and all the relative solutions have been found, thus we can ignore it and backtrack.

Algorithm 1: SCQ-ENUM

```
Input  : A graph G=(V,E)
Output : The set RESULT containing all maximal SCQs in G
Global : RESULT set, initially empty
1  for v ∈ V do
2      ENUM(COMPLETE({v}))
3  Function ENUM(S)
4      if contains(S, RESULT) then return
5      add(S, RESULT)
6      foreach x ∈ V \ S do
7          I ← (S ∩ N(x) ∪ {x})
8          X ← COMPLETE({x}, I)
9          if X = {x} then continue
10         ENUM(COMPLETE(X))
11     foreach {y, z} ⊆ V \ S do
12         I ← (S ∩ N(y) ∩ N(z) ∪ {y, z})
13         X ← COMPLETE({y, z}, I)
14         if X = {y, z} then continue
15         ENUM(COMPLETE(X))
```

4.2 Correctness

We will hereby prove the correctness of SCQ-ENUM. The principle of finding maximal solutions from other solutions is used by many enumeration algorithms, but this has so far been applied exclusively to properties with *strong accessibility* [2], such as hereditary [10, 14, 26], or connected-hereditary [4, 7].

Thanks to the results obtained in Section 3, however, we will be able to prove the correctness of our technique, despite SCQs not being strongly (or even weakly) accessible, similar to [7]. In the following, given two SCQs S and T, let $S \cap_{scq} T$ be the largest SCQ in $S \cap T$; we recall that this may be a single vertex, which is indeed an SCQ.

Proving that no solution is found twice by SCQ-ENUM is trivial, as duplication is removed by the RESULT set, and since every output is a maximal SCQ since it is the result of a COMPLETE call, we only need to prove that every maximal SCQ is found:

▶ **Theorem 9.** SCQ-ENUM *finds all and only maximal* SCQ*s exactly once.*

Proof. Let T be any solution not yet found by the algorithm. Let S be the solution found by SCQ-ENUM which maximizes $|S \cap_{scq} T|$. Note that $|S \cap_{scq} T| \geq 1$: for any $v \in T$, the algorithm will visit $C = \text{COMPLETE}(\{v\})$, a maximal SCQ containing v, so $|C \cap_{scq} T| \geq 1$. Now let $Z = S \cap_{scq} T$. We have $Z \neq T$, otherwise T would not be maximal, and by Lemma 2 there exists $Y \subseteq T \setminus Z$ with $1 \leq |Y| \leq 2$ s.t. $Z \cup Y$ is an SCQ. Note that Y is not contained in S, as otherwise $Z \cup Y$ would be a larger SCQ in $S \cap_{scq} T$. Three cases are possible: (i) $Y = \{x\}$, then $x \in V \setminus S$ and x is considered in the first *for* loop. (ii) $Y = \{y, z\} \subseteq V \setminus S$, then $\{y, z\}$ is considered in the second *for* loop. (iii) $|Y| = \{y, z\}$, with $y \in V \setminus S$ and $z \in S$, then y is considered in the first *for* loop and we will have $z \in S \cap N(y) \cup \{y\}$.

In all these cases, the SCQ X (maximal in I) that is found, will contain $Z \cup Y$ by Lemma 4. When we execute COMPLETE(X), we will either find T, or a maximal SCQ S' that contains

Algorithm 2: COMPLETE(X, A)

Input : X, an SCQ, and $A \subseteq V$, a set of vertices
Output : $X' \supseteq X$, an SCQ maximal with respect to A

```
Function COMPLETE(X, A)
    while EXT ← MIN-EXTENSION(X, A) ≠ null do
        X ← X ∪ EXT;
    return X;
Function MIN-EXTENSION(X, A)
    ADD ← {Y ⊆ A \ X : 1 ≤ |Y| ≤ 2 and X ∪ Y is an SCQ};
    return MIN-LEX(ADD)
```

$Z \cup Y$. As $|Z \cup Y| > |Z|$, we have $|S' \cap_{scq} T| > |S \cap_{scq} T|$. By induction, when visiting S' we will either find T, or S'' such that $|S'' \cap_{scq} T| > |S' \cap_{scq} T|$, until eventually, in at most $|T|$ such steps, SCQ-ENUM will find T. ◀

5 Complexity Analysis

We now analyze the complexity of SCQ-ENUM, *i.e.*, Algorithm 1, showing that it lists maximal SCQs with delay $O(\min(\omega(G)d^2\Delta^2, m^2))$. Recall that m, n, Δ, d and $\omega(G)$ are respectively the number of edges, number of vertices, maximum vertex degree, degeneracy and maximum size of an SCQ in G, and that the vertices are $v_1, \ldots, v_n$ are given in a *degeneracy ordering*. Firstly, we bound the complexity of the function COMPLETE:

▶ **Lemma 10.** COMPLETE(X, A) *(Algorithm 2) can be executed in time* $O(\min(d\Delta, m))$.

Proof. Consider the vertices in A adjacent to all vertices of X, *i.e.* $\bigcap_{x \in X}(N(x)) \cap A$, and partition them in three sets, each stored in increasing lexicographical order: SINK contains all the *sinks w.r.t.* X; SOURCE the *sources w.r.t.* X, and BOTH all vertices that have at least one edge *from* and one *towards* some vertex in X (*i.e.*, neither sinks nor sources *w.r.t.* to X). The computing time is the sum of the degrees of vertices in X, *i.e.* $\min(\omega(G)\Delta, m)$ and the total size $|\text{SINK}| + |\text{SOURCE}| + |\text{BOTH}|$ of the sets is bounded by Δ.

As each step adds either one or two vertices to X, and $|X|$ is bounded by $\omega(G)$, we will have at most $\omega(G)$ steps. Whenever we add a vertex a to X, we can update the SINK, SOURCE, BOTH sets by only looking at $N(a)$: any non neighbour of a is excluded from these sets, any vertex in SINK with an edge towards a, or vertex in SOURCE with an edge from a is moved in BOTH. This takes $O(|N(a)|)$ time, thus $O(\min(\omega(G)\Delta, m))$ time for all updates.

If the BOTH set is not empty, we can find the (lexicographically) smallest x in $O(1)$ time. Then, we need to find the smallest pair $a \in \text{SINK}, b \in \text{SOURCE}$ s.t. there is an edge from a to b, if it exists. We do this by scanning vertices in SINK $\cup$ SOURCE in order, and for each scanning its *forward* neighbours, still in order; we stop at the first pair that verifies the property. We never need to consider the same pair twice, as the condition (edge from a to b) will stay false (although the vertices might be later moved to BOTH and still enter X), thus the total cost will be $O(\min(d\Delta, m))$ for all steps. Finally, the smallest among x and $\{a, b\}$ corresponds to MIN-EXTENSION(X, A); we add it to X and update the SINK, SOURCE, BOTH sets. The total cost is given by $O(\min(\omega(G)\Delta, m) + \min(d\Delta, m)) = O(\min(d\Delta, m))$ ◀

Finally, we are ready to give the time complexity of SCQ-ENUM:

▶ **Lemma 11.** SCQ-ENUM *(Algorithm 1) has* $O(\min(\omega(G)d^2\Delta^2, m^2))$ *time delay.*

Proof. Let us first focus on the amortized cost per solution, *i.e.*, the cost of an execution of ENUM(S) without considering children recursive calls (which lead to other solutions). To compute *contains*$(S,$ RESULT$)$ and *add*$(S,$ RESULT$)$ we store RESULT as a *trie*, whose depth will be $O(\omega(G))$, and degree will be bounded by Δ as SCQs are made of adjacent vertices.[5] Checking the existence and adding a solution to this trie takes time $O(\min(\omega(G)\log(\Delta), m))$.

In the *for* loops we only need to consider x and $\{y, z\}$ s.t. $I \neq \{x\}$ and $I \neq \{y, z\}$, thus only vertices with a neighbour in S: for x we have $|S|\Delta \leq \min(\omega(G)\Delta, n)$ choices; as for $\{y, z\}$ we have $\min(\omega(G)\Delta, n)$ choices for y, and for each y up to d choices for z (we only need to consider each pair once, *e.g.* when $y < z$, so we only scan the *forward* neighbours of y), for a total of $\min(\omega(G)d\Delta, m)$ choices. The cost of each loop is given by $O(\Delta)$ for computing I, and $O(\min(d\Delta, m))$ to compute X and COMPLETE(X). If the recursive call ENUM(COMPLETE(X)) will generate a new solution, the cost will be attributed to the child solution; otherwise the recursive call will only perform the *contains*$(S,$ RESULT$)$ call.

The total cost of an iteration of ENUM(S) is thus the cost of the *contains*/*add* procedures, plus the number of execution of the loops times the cost of a loop iteration, *i.e.*, $O(\min(\omega(G)\log(\Delta), m) + \min(\omega(G)d\Delta, m) \cdot (\min(\omega(G)\log(\Delta), m) + \min(d\Delta, m)))$. Thus, the cost per solution is bounded by both $O(\omega(G)d^2\Delta^2)$ and $O(m^2)$.

Finally, as each recursive call outputs a solution, we can exploit the alternative output method by Uno [27], as done in [20, 10]: we output a solution at the beginning of a recursive call when the depth of the recursion tree is even, and at the end when it is odd; this way the delay of the algorithm will be equal to the amortized cost per solution. ◀

Calling α the number of solutions, this gives us a total time of $\alpha \cdot O(\min(\omega(G)d^2\Delta^2, m^2))$. The space complexity is dominated by the size of the RESULT set, that is $O(\alpha \cdot \omega(G))$ as it will contain α solutions of size bounded by $\omega(G) \leq d + 1 \leq n$. While α is potentially exponential, we remark that SCQ-ENUM can still be efficiently applied to analyze real world networks: recalling Corollary 5 and the discussion in Section 3.3, we have that α will only be up to a factor $\omega(\mathsf{u}(G)) \leq n$ larger than the number of maximal (undirected) cliques in $\mathsf{u}(G)$. It is generally agreed upon that real-world networks are sparse [12, 15], and as such contain an extremely small number of maximal cliques compared to the theoretical maximum [24]. Furthermore, the number of maximal cliques is actually polynomial when the degeneracy (or arboricity) is bounded [12], which is the case in many sparse networks.

6 Conclusions and Future Work

In this work we proposed a model for communities in directed graphs, that of maximal strongly connected cliques. We analyzed this model, giving tight bounds on the number of such cliques in an n-vertex graph and proving some accessibility properties. We exploited these properties to produce SCQ-ENUM, an algorithm that lists maximal strongly connected cliques with polynomial delay, *i.e.*, $O(\min(\omega(G)d^2\Delta^2, m^2))$, that can be a valid tool for analyzing the community structures of directed real-world networks.

Future work is focused in two directions: the first is using the proposed algorithm to discover new community structures in directed networks, while the second is to further investigate the generalized definition of accessibility given by the existence of an addible (or removable) set of elements of bounded size to each non-maximal solution.

[5] The root of the trie has degree up to n; we store this as a vector of size n, accessible in $O(1)$ time.

References

1 Eralp Abdurrahim Akkoyunlu. The enumeration of maximal cliques of large graphs. *SIAM J Comput*, 2(1):1–6, 1973.

2 Hiroki Arimura and Takeaki Uno. Polynomial-delay and polynomial-space algorithms for mining closed sequences, graphs, and pictures in accessible set systems. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 1088–1099. SIAM, 2009.

3 Baruch Awerbuch and Yuval Shavitt. Topology aggregation for directed graphs. *IEEE/ACM Transactions On Networking*, 9(1):82–90, 2001.

4 Devora Berlowitz, Sara Cohen, and Benny Kimelfeld. Efficient enumeration of maximal k-plexes. In *SIGMOD*, pages 431–444, New York, NY, USA, 2015. ACM.

5 Coenraad Bron and Joep Kerbosch. Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM*, 16(9):575–576, 1973.

6 Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, pages 84–95. Springer, 2000.

7 Sara Cohen, Benny Kimelfeld, and Yehoshua Sagiv. Generating all maximal induced subgraphs for hereditary and connected-hereditary graph properties. *Journal of Computer and System Sciences*, 74(7):1147 – 1159, 2008.

8 Carlo Comin and Romeo Rizzi. An improved upper bound on maximal clique listing via rectangular fast matrix multiplication. *CoRR*, abs/1506.01082, 2015.

9 Alessio Conte, Roberto Grossi, Andrea Marino, Romeo Rizzi, and Luca Versari. Directing road networks by listing strong orientations. *IWOCA*, pages 83–95, 2016.

10 Alessio Conte, Roberto Grossi, Andrea Marino, and Luca Versari. Sublinear-space bounded-delay enumeration for massive network analytics: Maximal cliques. In *ICALP*, 2016.

11 Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics)*. Springer, 2005.

12 David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. *ACM Journal of Experimental Algorithmics*, 18, 2013.

13 Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

14 Komei Fukuda. Note on new complexity classes ENP, EP and CEP, 1996. Accessed: 02-2016. URL: `https://www.inf.ethz.ch/personal/fukudak/old/ENP_home/ENP_note.html`.

15 Gaurav Goel and Jens Gustedt. Bounded arboricity to determine the local structure of sparse graphs. In *WG*, pages 159–167. Springer, 2006.

16 Björn H. Junker and Falk Schreiber. *Analysis of biological networks*, volume 2. John Wiley & Sons, 2011.

17 Jon M. Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew S. Tomkins. The web as a graph: Measurements, models, and methods. In *International Computing and Combinatorics Conference*, pages 1–17. Springer, 1999.

18 Darong Lai, Hongtao Lu, and Christine Nardini. Finding communities in directed networks by pagerank random walk induced network embedding. *Physica A: Statistical Mechanics and its Applications*, 389(12):2443–2454, 2010.

19 Elizabeth A. Leicht and Mark E.J. Newman. Community structure in directed networks. *Physical review letters*, 100(11):118703, 2008.

20 Kazuhisa Makino and Takeaki Uno. New algorithms for enumerating all maximal cliques. In *Algorithm Theory-SWAT 2004*, pages 260–272. Springer, 2004.

21 John W. Moon and Leo Moser. On cliques in graphs. *Isr J Math*, 3(1):23–28, 1965.

22 Jeffrey Pattillo, Nataly Youssef, and Sergiy Butenko. Clique relaxation models in social network analysis. *Handbook of Optimization in Complex Networks*, pages 143–162, 2012.

23 Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *P Natl Acad Sci USA*, 105(4):1118–1123, 2008.

24 Matthew C. Schmidt, Nagiza F. Samatova, Kevin Thomas, and Byung-Hoon Park. A scalable, parallel algorithm for maximal clique enumeration. *J Parallel Distr Com*, 69(4):417–428, 2009.

25 John Scott. *Social network analysis*. Sage, 2012.

26 Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM J Comput*, 6(3):505–517, 1977.

27 Takeaki Uno. Two general methods to reduce delay and change of enumeration algorithms. *NII Technical Report NII-2003-004E, Tokyo, Japan*, 4, 2003.

28 Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.