Isolation Concepts for Efficiently Enumerating Dense Subgraphs ⋆

Christian Komusiewicz [1] Falk Hüffner [2] Hannes Moser [3]
Rolf Niedermeier

Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany

Abstract

In an undirected graph $G = (V, E)$, a set of k vertices is called c-isolated if it has less than $c \cdot k$ outgoing edges. Ito and Iwama [ACM Trans. Algorithms, to appear] gave an algorithm to enumerate all c-isolated maximal cliques in $O(4^c \cdot c^4 \cdot |E|)$ time. We extend this to enumerating all maximal c-isolated cliques (which are a superset) and improve the running time bound to $O(2.89^c \cdot c^2 \cdot |E|)$, using modifications which also facilitate parallelizing the enumeration. Moreover, we introduce a more restricted and a more general isolation concept and show that both lead to faster enumeration algorithms. Finally, we extend our considerations to s-plexes (a relaxation of the clique notion), providing a W[1]-hardness result when the size of the s-plex is the parameter and a fixed-parameter algorithm for enumerating isolated s-plexes when the parameter describes the degree of isolation.

Key words: NP-hard problem, parameterized complexity, fixed-parameter tractability, exact algorithm, clique, s-plex

1 Introduction

Finding and enumerating cliques and clique-like structures in graphs has many applications ranging from technical networks [14] to social and biological networks [2,3,13,18]. Unfortunately, clique-related problems are known to be notoriously hard for exact algorithms, approximation algorithms, and fixed-parameter algorithms. Consider for example the problem of finding a clique of maximum size:

> Maximum Clique
> **Input:** An undirected graph $G = (V, E)$ and a nonnegative integer k.
> **Question:** Is there a subset of vertices $S \subseteq V$ with $|S| \leq k$ such that all vertices in S are pairwise connected in G?

Maximum Clique has been shown to be NP-complete [10], inapproximable in polynomial time within a factor of $|V|^{1-\epsilon}$ [11], and W[1]-hard when parameterized with the size k of the maximum clique [6]. Ito et al. [15] introduced an interesting way out of this quandary by restricting the search to *isolated* cliques. Herein, a clique is seen as isolated if the vertices in the clique have few neighbors outside of the clique. This is formally described via the concept of *c-isolation*:

Definition 1 *Let $G = (V, E)$ be an undirected graph. A set $S \subseteq V$ of k vertices is called* c-isolated *if it has less than $c \cdot k$ outgoing edges, where an outgoing edge is an edge between a vertex in S and a vertex in $V \setminus S$.*

⋆ A preliminary version of this paper appeared in the proceedings of the 13th International Computing and Combinatorics Conference (COCOON '07), held in Banff, Canada, July 16–19 [17]. Fundamental parts of this work together with related topics also appeared in the first author's diploma thesis [16]. Note that in our conference version [17] we state that the work by Ito et al. [15], which is the conference version of Ito and Iwama [14], is flawed. A reason for this statement was a misunderstanding of Ito et al.'s [15] enumeration concept: they enumerate all isolated maximal cliques, and not all maximal isolated cliques, as we were assuming.

Email addresses: c.komus@uni-jena.de (Christian Komusiewicz), hueffner@minet.uni-jena.de (Falk Hüffner), hannes.moser@uni-jena.de (Hannes Moser), rolf.niedermeier@uni-jena.de (Rolf Niedermeier).

[1] Partially supported by the Deutsche Forschungsgemeinschaft, project OPAL (optimal solutions for hard problems in computational biology), NI 369/2, and a PhD fellowship of the Carl-Zeiss-Stiftung.

[2] Supported by the Deutsche Forschungsgemeinschaft, Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4, and the Edmond J. Safra Foundation.

[3] Supported by the Deutsche Forschungsgemeinschaft, project ITKO (iterative compression for solving hard network problems), NI 369/5, and project AREG (algorithms for generating quasi-regular structures in graphs), NI 369/9.

As their main result, Ito and Iwama [14] gave an $O(4^c \cdot c^4 \cdot |E|)$ time algorithm for enumerating all c-isolated maximal cliques in a graph. In particular, this means linear time for constant c and fixed-parameter tractability with respect to the parameter c.

Instead of c-isolated maximal cliques (that is, cliques that are c-isolated and not contained in any other clique), we enumerate maximal c-isolated cliques (that is, cliques that are c-isolated and not contained in any other *c-isolated* clique). Clearly, our approach produces a superset of cliques. Using a postprocessing step, we can easily obtain just the c-isolated maximal cliques, if desired (in fact, our algorithms can be modified to directly produce this result). However, the enumeration of maximal c-isolated cliques has some potential advantages: it produces "interesting" isolated cliques that the enumeration of c-isolated maximal cliques misses. Consider for example a clique of size four, where one vertex additionally has seven degree-1 neighbors. It contains no 2-isolated maximal clique, but one maximal 2-isolated clique. More generally, adding a *hub* (vertex connected to all other vertices) to an n-vertex graph destroys all c-isolated maximal cliques of size up to $n/(c+1)$, and adding a hub attached to $c \cdot n$ degree-1 vertices even destroys all c-isolated maximal cliques. Since hubs and small-degree vertices are a common feature of real-world networks such as scale-free networks, this might limit the usefulness of enumerating maximal c-isolated cliques in these cases. In contrast, when enumerating c-isolated maximal cliques, hub vertices can just be omitted from enumerated cliques, leading to potentially more useful results.

Our paper is organized as follows. Section 2 deals with the enumeration of c-isolated maximal cliques as proposed by Ito and Iwama [14]. We start with describing their algorithm, and then show how to adapt it to enumerate maximal c-isolated cliques instead of c-isolated maximal cliques. We further modify the algorithm to improve the asymptotic running time bound to $O(2.89^c \cdot c^2 \cdot |E|)$. Moreover, this allows to parallelize the so far purely sequential enumeration algorithm. [4] Note that in the rest of the paper we assume for simplicity that the isolation factor c is an integer.

In Section 3, inspired by Ito and Iwama's isolation concept, we propose two further isolation definitions, the weaker (less demanding) *min-c-isolation* and the stronger *max-c-isolation*, both practically motivated. Whereas c-isolation makes restrictions on the *sum* of outgoing edges, we now restrict the number of outgoing edges for *at least one* vertex (min-c-isolation) or for *all* vertices (max-c-isolation) of a vertex set to be less than c. Figure 1 gives an example for each isolation concept with $c = 2$. Somewhat surprisingly, we can show that—compared to c-isolation—*both* concepts lead to faster enumeration algorithms

[4] Note that in the conference version [17] of our paper a weaker bound of 4^c instead of 2.89^c was given.

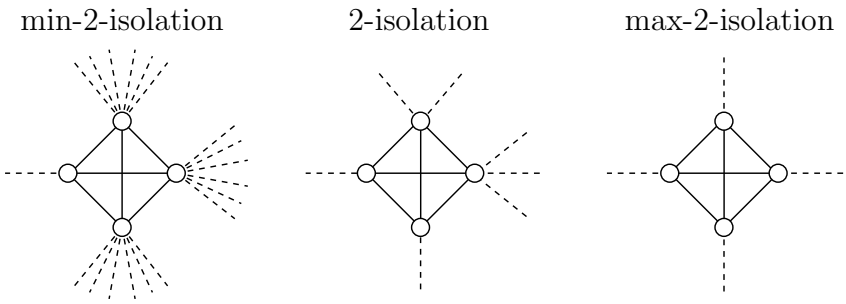

Fig. 1. Example isolated sets for $c = 2$ and the three isolation concepts. In a min-c-isolated vertex set, *at least one* vertex has less than c outgoing edges, in a c-isolated vertex set S the *sum of outgoing edges* is less than $c \cdot |S|$, and in a max-c-isolated vertex set *every* vertex has less than c outgoing edges.

Table 1
Summary of the running times of the enumeration algorithms for isolated dense subgraphs for n-vertex and m-edge graphs.

Enumeration Task	Running Time
c-isolated cliques	$O(2.89^c \cdot c^2 \cdot m)$
min-c-isolated cliques	$O(2^c \cdot c \cdot m + n \cdot m)$
max-c-isolated cliques	$O(2.44^c \cdot c \cdot m)$
min-c-isolated s-plexes	$O((s+1)^c \cdot (s+c) \cdot n^{s+1} + n \cdot m)$

for isolated cliques, improving the exponential factor from 2.89^c to 2^c and 2.44^c, respectively.

In Section 4, we show how to adapt the isolation scenario to the concept of s-plexes, a relaxation of cliques occurring, for example, in social network analysis [25,1,18]. In an undirected graph $G = (V, E)$, a vertex subset $S \subseteq V$ of size k is called an *s-plex* if the minimum degree in $G[S]$ is at least $k - s$. Hence, cliques are exactly 1-plexes. First, strengthening a previous NP-hardness result [1,18,19], we point out that the problem of finding s-plexes is W[1]-hard with respect to the parameter k; that is, the problem seems as (parameterized) intractable as Maximum Clique is. This motivates the consideration of isolation as a parameter for s-plex enumeration. We show that we can indeed apply the isolation scenario for s-plexes by providing a fixed-parameter algorithm with parameter c that enumerates all maximal min-c-isolated s-plexes for constant s. As a side result, we improve a time bound for a generalized vertex cover problem first studied by Nishimura et al. [22]. Table 1 gives a summary of the running times of the enumeration algorithms for combinations of isolation conditions and dense subgraphs that are considered in this work.

Preliminaries. We only consider undirected graphs $G = (V, E)$ with $n := |V|$ and $m := |E|$. Let $N(v) := \{u \in V \mid \{u, v\} \in E\}$ and $N[v] := N(v) \cup \{v\}$. For $v \in V$, let $\deg(v) := |N(v)|$. For $A, B \subseteq V, A \cap B = \emptyset$, let $E(A, B) :=$

$\{\{u, v\} \in E \mid u \in A, v \in B\}$. For a vertex set $S \subseteq V$, an *outgoing edge* is an edge $\{u, v\} \in E(S, V \setminus S)$, and for a vertex $v \in S$, its outgoing edges are the outgoing edges of S that are incident on v. For $V' \subseteq V$, let $G[V']$ be the subgraph of G induced by V' and $G \setminus V' := G[V \setminus V']$. For $v \in V$, let $G - v := G[V \setminus \{v\}]$. The *complement graph* $\bar{G} = (V, \bar{E})$ of a graph $G = (V, E)$ is defined as the graph with the same set of vertices and complementary edge set $\bar{E} := \{\{u, v\} \subseteq V \mid \{u, v\} \notin E\}$. A set S with property P is called *maximal* if no proper superset of S has property P, and *maximum* if no other set with property P has higher cardinality. A *clique* is a set of vertices $S \subseteq V$ that induces a complete graph. A *vertex cover* of a graph G is a set of vertices $S \subseteq V$ such that every edge in G has at least one of its endpoints in S.

Parameterized complexity [6,8,21] is an approach to finding optimal solutions for NP-hard problems [10]. The idea is to accept the seemingly inevitable combinatorial explosion, but to confine it to one aspect of the problem, the *parameter*. If for relevant inputs this parameter remains small, then even large problems can be solved efficiently. More precisely, a problem is *fixed-parameter tractable* (FPT) with respect to a parameter k if there is an algorithm solving a problem instance of size n in $f(k) \cdot n^{O(1)}$ time for some computable function f. Analogously to NP-hardness, one can show that a problem is presumably *not* fixed-parameter tractable by giving a parameterized reduction from a problem known to be in the class W[1], thus showing *W[1]-hardness*. A *parameterized reduction* reduces an instance (I, k) of a parameterized problem, where k is the parameter, in $f(k) \cdot |I|^{O(1)}$ time for an arbitrary computable function f to an instance (I', k') of another parameterized problem, such that (I, k) is a yes-instance iff (I', k') is a yes-instance and k' only depends on k, not on $|I|$.

2 Enumerating Isolated Cliques

We begin with describing Ito and Iwama's [14] algorithm for enumerating c-isolated maximal cliques. Given a graph $G = (V, E)$ and an isolation factor c, first the vertices are sorted by their degree such that $u < v \Rightarrow \deg(u) \leq \deg(v)$. The *index* of a vertex is its position in this sorted order. Let $N_+[v] := \{u \in N[v] \mid u > v\} \cup \{v\}$ and $N_-(v) := \{u \in N(v) \mid u < v\}$.

In a c-isolated clique, the vertex with the lowest index is called the *pivot* of the clique. Clearly, a pivot has less than c outgoing edges. Since every c-isolated clique has a pivot, we can enumerate all c-isolated maximal cliques of a graph by enumerating for each $v \in V$ all c-isolated maximal cliques in $G[N_+[v]]$ (comprising all c-isolated maximal cliques with pivot v) and then removing those c-isolated cliques that are a subset of a c-isolated clique with another pivot $u \in N_-(v)$.

The enumeration of c-isolated maximal cliques with pivot v for each $v \in V$ is the central part of the algorithm. We call this the *pivot procedure*. It comprises three successive stages.

Trimming stage. In this stage, one builds a candidate set C that is a superset of all c-isolated cliques with pivot v. The candidate set C is initialized with $N_+[v]$, and then vertices that obviously cannot be part of a c-isolated clique with pivot v are removed from C. The stage provides the following invariants for each $u \in C$, which help in upper-bounding the running time of later stages:

(a) $\deg(u) < (c+1) \cdot |C| - 1$;
(b) u has fewer than $c \cdot |C|$ outgoing edges;
(c) u has at least $|C| - c$ neighbors in C; and
(d) for any $h \leq |C|$, C^h has less than $c \cdot (c+1) \cdot h$ outgoing edges, where C^h denotes the set of the h vertices with the h lowest indices in C.

Establishing invariant (a) can be done in $O(\deg(v))$ time. Afterwards all vertices in C have bounded degree, which reduces the time spend during the scans of the adjacency lists that have to performed to establish invariants (b), (c) and (d). For details on correctness and analysis we refer to Ito and Iwama [14].

Enumeration stage. In this stage, cliques with pivot v are enumerated. Let C be the candidate set after the trimming stage, and $|N[v] \setminus C| = d$. In total, we can delete at most $c-1$ vertices from $N[v]$, since otherwise v obtains too many outgoing edges. Therefore, $\tilde{c} := c - 1 - d$ is the number of vertices that we may still remove from C. We can enumerate *maximal* cliques $C' \subseteq C$ of size *at least* $|C| - \tilde{c}$ by enumerating *minimal* vertex covers of size *at most* $\tilde{c}$ in the complement graph $\overline{G[C]}$.

Screening stage. In the screening stage, all cliques that are either not c-isolated or that are c-isolated but not maximal are removed. First, c-isolation is checked. Then, we check each clique that is left for pivot v against each clique obtained during calls to pivot(u) with $u \in N_-(v)$, since these are the only cliques that can be superset of a clique obtained for pivot v. The overall running time, in the exponential part dominated by this last step, is then $O(4^c \cdot c^4 \cdot |E|)$ [14].

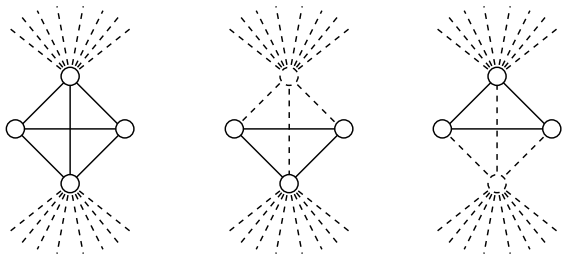

Fig. 2. Example for a maximal clique that is not 4-isolated (a) but has two subsets that are 4-isolated ((b) and (c)). *Solid lines* are edges between members of the clique; *dashed lines* are outgoing edges.

2.1 Enumerating Maximal c-Isolated Cliques

In this section, we show how the algorithm by Ito and Iwama [14] can be modified in order to enumerate not only c-isolated maximal cliques but also maximal c-isolated cliques. Figure 2 shows that a maximal clique that violates the isolation condition might have several subsets that *are* c-isolated. The clique in Figure 2 (a) has four vertices and 16 outgoing edges and is thus not 4-isolated. However, two subsets ((b) and (c) in Figure 2) are cliques with three vertices and 11 outgoing edges, and thus are 4-isolated.

Our algorithm follows a two-step approach. First, we enumerate all *minimal* vertex covers and thus obtain *maximal* cliques in the candidate set C. Then, to also capture c-isolated cliques that are subsets of non-c-isolated cliques enumerated this way, for each of these cliques, we enumerate all maximal subsets that fulfill the isolation condition. The problem of finding these c-isolated subsets of a clique can be stated as follows:

Isolated Clique Subsets
Input: A graph $G = (V, E)$, a clique $C \subseteq V$ and a nonnegative integer c.
Task: Find all maximal sets $C' \subseteq C$ that form a c-isolated clique, that is, a clique with less than $c \cdot |C'|$ outgoing edges.

The difficulty lies in solving Isolated Clique Subsets with the running time depending only polynomially on $|C|$. We will achieve this by showing in Lemma 3 that only vertices of high index need to be considered for removal. The following two lemmas are needed for the proof of Lemma 3.

Lemma 1 *Let C be a clique of size k. Then, any c-isolated subset $C' \subset C$ has size at least $k - c + 1$.*

PROOF. Every vertex in a set $C' \subset C$ is adjacent to all vertices in $C \setminus C'$. Hence, if C' has less than $k - c + 1$ members, every vertex is adjacent to c

or more vertices outside of C' and thus C' has more than $c \cdot |C'|$ outgoing edges. $\square$

Lemma 2 *Let C be a clique of size k. Then, any maximal c-isolated subset $C' \subset C$ of size d contains those vertices that have at most d neighbors in $V \setminus C$.*

PROOF. We show the contraposition: Let C' be a c-isolated subset of C of size d; if there is a vertex $v \in C \setminus C'$ that has at most d neighbors in $V \setminus C$, then C' is not maximal.

Comparing the number of outgoing edges of C' (which are at most $cd-1$) with the number of outgoing edges of $C' \cup \{v\}$, we can observe that each vertex in C' has one less outgoing neighbor (because v is not "outside" anymore), and v has at most $|C \setminus C'| + d = (k-d) + d = k$ outgoing edges. Therefore, the number of outgoing edges of $C' \cup \{v\}$ is at most $(cd-1) - d + k$, which is at most $c \cdot (d+1) - 2$ by using the inequality $k \leq d + c - 1$ due to Lemma 1. Thus, $C' \cup \{v\}$ has less than $c \cdot |C' \cup \{v\}|$ outgoing edges and is therefore c-isolated and C' consequently is not a maximal c-isolated subset of C. $\square$

With these two lemmas, we have the prerequisites to show that the vertices with the $k-c+1$ lowest indices must belong to any maximal c-isolated subset of C.

Lemma 3 *Let C be a clique of size k. Every maximal c-isolated subset of C is a superset of C^{k-c+1}, where C^{k-c+1} is the set of vertices with the $k-c+1$ lowest indices in C.*

PROOF. Let C' be a maximal c-isolated subset of C that is not a superset of C^{k-c+1}. We show that C' is not c-isolated, leading to a contradiction. Let $\tilde{c} := |C \setminus C'|$ be the number of vertices that were removed from C. All vertices in C' are adjacent to all vertices in C. Hence, there are $(k-\tilde{c}) \cdot \tilde{c}$ edges between C' and C. Moreover, consider a vertex $u \in C^{k-c+1} \setminus C'$. Such a vertex must exist since $C^{k-c+1} \not\subseteq C'$. Clearly, there are at least $c-1-(\tilde{c}-1) = c-\tilde{c}$ vertices in C' that have higher index than u. Since C' is a maximal c-isolated clique, u must have more than $k-\tilde{c}$ neighbors in $V \setminus C$ (as shown by Lemma 2). Hence, each of the $c-\tilde{c}$ vertices in C' that have higher index than u has more than $k-\tilde{c}$ neighbors in $V \setminus C$. The number of outgoing edges $x(C')$ from C' thus is

$$x(C') > (k-\tilde{c}) \cdot \tilde{c} + (c-\tilde{c}) \cdot (k-\tilde{c}) = c \cdot (k-\tilde{c}) = c \cdot |C'|.$$

Therefore, C' is not c-isolated, thus proving the claim. $\square$

	procedure isolated-subset(C, c)
Input:	A clique $C = \{v_1, v_2, \ldots, v_k\}$ with vertices sorted by degree and a nonnegative integer c.
Output:	The set $\mathcal{C}$ of maximal c-isolated cliques in C.
1:	$e(C) := (\sum_{v \in C} \deg(v)) - c \cdot \lvert C \rvert + 1$
2:	$\mathcal{D} := \{\emptyset\}$, $\mathcal{C} := \emptyset$
3:	**repeat** c **times**
4:	**foreach** $D \in \mathcal{D}$
5:	**if** $C \setminus D$ is a c-isolated clique **then** $\mathcal{C} := \mathcal{C} \cup \{C \setminus D\}$
6:	**else**
7:	**if** $D = \emptyset$ **then** $i := k + 1$ **else** $i := \min_{v_l \in D}\{l\}$
8:	$\mathcal{D} := \mathcal{D} \cup \{D \cup \{v_j\} \mid k - c < j < i\}$
9:	$\mathcal{D} := \mathcal{D} \setminus \{D\}$
10:	**return** $\mathcal{C}$

Fig. 3. Algorithm for enumerating maximal c-isolated subsets of a clique C

Lemma 3 provides the basis for an enumeration algorithm for Isolated Clique Subsets. Mainly, this algorithm generates subsets of $C \setminus C^{k-c+1}$ in order of increasing cardinality and tests whether removal of these subsets yields a c-isolated subset of C. We ensure that no superset of a set whose removal yields a c-isolated clique is enumerated and thus we only output maximal c-isolated subsets of C.

Theorem 1 *Any instance of* Isolated Clique Subsets *has at most* 2^c *solutions, and they can be enumerated in* $O(2^c + |C|)$ *time.*

PROOF. We describe an algorithm for Isolated Clique Subsets and bound the number of enumerated subsets. The pseudo code of the algorithm is shown in Figure 3. It enumerates minimal vertex sets that we need to remove in order to restore the isolation condition and outputs the maximal cliques obtained after the removal of these vertices. In line 1, we compute the number of excessive outgoing edges $e(C)$ from C, that is, the number of outgoing edges above the threshold allowed by the isolation condition. This is done because then we can check in constant time in line 5 whether the isolation condition is fulfilled by keeping and updating the number of excessive outgoing edges for each deletion candidate set.

In the repeat loop of line 3 we create the candidate deletion sets in order of increasing cardinality. In the q-th pass of this repeat loop, the set $\mathcal{D}$ contains all deletion candidate sets of size $q - 1$, except those that are superset of a deletion set whose removal yields a c-isolated clique.

For each set $D \in \mathcal{D}$ we test in line 5 whether the deletion of the vertices in D yields a c-isolated clique. If this is the case, then $C \setminus D$ is added to

the set $\mathcal{C}$ of maximal c-isolated cliques and no supersets of D are considered as deletion candidate sets, because they would yield non-maximal c-isolated cliques. Otherwise, we insert supersets of D of size $|D| + 1$, which are then tested in the next pass of the repeat-loop. By Lemma 3, we must not remove vertices that belong to $C^{|C|-c+1}$. Consequently, we only create supersets of D that contain vertices from $C \setminus C^{|C|-c+1}$. We also do not add vertices whose index is larger than the smallest index of a vertex in D. This makes sure that no set is generated twice and that we only create supersets of sets whose removal does not yield a c-isolated clique. Afterwards, we remove D from our set of candidate sets (line 9).

The computation of excessive outgoing edges can be done in $O(|C|)$ time. We start with the empty set as only candidate set. The number of excessive outgoing edges of this candidate set is exactly the number of outgoing edges from C. Whenever we create a superset $D' = D \cup \{v\}$ of a candidate set D, the number of excessive outgoing edges from $C \setminus D'$, $e(C \setminus D') = e(C \setminus D) - \deg(v) + 2 \cdot |C \setminus D'| + c$, can be computed in $O(1)$ time. Thus we need $O(1)$ time for each enumerated subset. Clearly, at most 2^c subsets are enumerated, since we only consider vertices from $C^{|C|-c+1}$. Hence, we can enumerate all c-isolated subsets of a clique C in $O(2^c + |C|)$ time. □

Before actually solving an instance of Isolated Clique Subsets for an enumerated clique, we make use of the fact that we can transform instances with isolation factor c in which every vertex has at least $x_{\min}$ outgoing edges into instances with isolation factor $c - x_{\min}$. This will help in upper-bounding the running time of our modified pivot procedure (see Proposition 1).

Lemma 4 *Let $L = (G, C, c)$ be an instance of* Isolated Clique Subsets *in which every vertex $v \in C$ has at least $x_{\min}$ outgoing edges from C. We can transform L into an equivalent instance L' with isolation factor set to $\hat{c} := c - x_{\min}$.*

PROOF. Let L' be a transformed instance of Isolated Clique Subsets that we obtain after applying the following two transformation rules to L:

(1) For each vertex $v \in C$, set $\deg(v) := \deg(v) - x_{\min}$;
(2) set $c := c - x_{\min}$.

Let C' be a subset of C. We show that C' is a member of the solution to L iff it is also a member of the solution to L'. For a problem instance L and a vertex set C', let $e(C', L)$ be the number of excessive outgoing edges, that is, the number of outgoing edges above the threshold defined by the isolation

condition. Clearly, the following equivalence holds:

$$\begin{aligned} e(C', L) &= \sum_{v \in C'} (\deg(v) - (|C'| - 1)) - c \cdot C' + 1 \\ &= \sum_{v \in C'} (\deg(v) - x_{\min} - (|C'| - 1)) - (c - x_{\min}) \cdot C' + 1 \\ &= e(C', L'). \end{aligned}$$

Since the number of excessive outgoing edges of a vertex set in the original instance L and in the transformed instance L' are equal, every vertex set C' is a member of the solution to L iff it is a member of the solution to L'. □

We now describe how to use Theorem 1 to obtain a pivot procedure for enumerating maximal c-isolated cliques. Our modified pivot procedure differs from the original procedure only in the enumeration stage and in the screening stage, not in the trimming stage. In the enumeration stage, we first build the complement graph $\overline{G[C]}$ as in the original pivot procedure. The enumeration of cliques is divided into two steps: the enumeration of maximal cliques and the enumeration of maximal subsets that fulfill the isolation condition for each of those cliques.

In the screening stage, we need not test the enumerated cliques for c-isolation, since this is already done in the *isolated-subset* procedure. However, we still have to filter non-maximal cliques. In the following proposition, we bound the running time of the modified enumeration algorithm.

Proposition 1 *All maximal c-isolated cliques of an m-edge graph $G = (V, E)$ can be enumerated in $O(4^c \cdot c^3 \cdot m)$ time.*

PROOF. Ito and Iwama [14] showed that the time spent during the trimming stage of all pivot procedures is $O(c^3 \cdot m)$. Let C_v be the candidate set for pivot v after the trimming stage, and let $m(C_v)$ be the number of edges in $G[C_v]$. After the trimming stage, there are at most $(c-1) \cdot |C_v|$ edges missing from C_v (due to invariant (c)). This means that the construction of the complement graph $\overline{G[C_v]}$ can be performed in $O(m(C_v) + c \cdot |C_v|)$ time, since $O(|C_v|^2) = O(m(C_v) + c \cdot |C_v|)$. After this, we first enumerate maximal cliques and then enumerate maximal c-isolated subsets of these cliques. The first enumeration step enumerates at most 2^{c-1} cliques and can be executed in $O(2^c \cdot c^2 + m(C_v))$ time [5], via the enumeration of minimal vertex covers in $\overline{G[C_v]}$. The running time of the second step depends on the size and number of vertex covers that were enumerated during the first step. We split the running time analysis of each call to the *isolated-subset* procedure (Figure 3) into the analysis of the time needed for the computation of the number of excessive outgoing edges (line 1) and the time needed for the enumeration of isolated

subsets (lines 3–9). At most 2^{c-1} vertex covers have been enumerated in the first step of the enumeration stage, and we spend $O(|C_v|)$ time for the computation of the number of excessive outgoing edges for each of them. Overall the computation of excessive outgoing edges thus takes $2^{c-1} \cdot O(|C_v|) = O(2^c \cdot |C_v|)$ time.

The time we spend for the enumeration of maximal subsets depends on the size of the enumerated vertex covers. Suppose that we have enumerated a vertex cover of size $\tilde{c}$. Then the resulting clique has size $|C_v| - \tilde{c}$. Since v is the vertex of minimum degree in that clique and $\deg(v) \geq |C_v|$, the minimum number of outgoing edges from each vertex in that clique is at least $\tilde{c}$. According to Lemma 4, we can transform the instance of Isolated Clique Subsets into an instance of Isolated Clique Subsets with isolation factor set to $c - \tilde{c}$. By Theorem 1, the enumeration of maximal subsets in *isolated-subsets* then can be performed in $O(2^{c-\tilde{c}})$ time. There are at most $2^{\tilde{c}}$ minimal vertex covers of size $\tilde{c}$, so we can enumerate the isolated subsets of all cliques of size $|C_v| - \tilde{c}$ in $O(2^{\tilde{c}} \cdot 2^{c-\tilde{c}}) = O(2^c)$ time. In total, the enumeration of maximal subsets in the *isolated-subsets* procedure takes

$$\sum_{\tilde{c}=0}^{c-1} O(2^c) = O(2^c \cdot c)$$

time. Note that for the same reason $2^c \cdot c$ is also an upper bound for the number of cliques that are enumerated during one execution of the enumeration stage. Overall, the running time for the enumeration stage in the pivot procedure for pivot v amounts to

$$O(m(C_v) + c \cdot |C_v| + 2^c \cdot c^2 + m(C_v)) + O(2^c \cdot |C_v| + 2^c \cdot c) = O(2^c \cdot c^2 \cdot m(C_v) + c \cdot |C_v|).$$

Ito and Iwama [14] also showed that $\sum_{v \in V} m(C_v) = O(c^3 \cdot m)$. Furthermore, $|C_v| \leq \deg(v)$ which leads to a total running time of the enumeration stages of all pivot procedures of

$$\sum_{v \in V} O(2^c \cdot c^2 \cdot m(C_v) + c \cdot \deg(v)) = O(2^c \cdot c^5 \cdot m).$$

In the screening stage for pivot v, each enumerated clique has to be compared to the cliques that were enumerated during the enumeration stage for the same pivot and to the cliques that were enumerated for pivots $u \in N_-(v)$. Since at most $2^c \cdot c$ cliques are enumerated during one execution of the enumeration stage, and $|N_-(v)| < c$ (otherwise there is no c-isolated clique with pivot v), we have to perform at most

$$O(2^c \cdot c \cdot 2^c \cdot c) + O(2^c \cdot c \cdot 2^c \cdot c^2) = O(4^c \cdot c^3)$$

pairwise comparisons. Since the cliques obtained in the enumeration stage have size at most $\deg(v)$, these comparisons can be performed in $O(4^c \cdot c^3 \cdot \deg(v))$

time. Together with the running times of the trimming and enumeration stages we can upper-bound the running time of the whole algorithm:

$$O(c^3 \cdot m) + O(2^c \cdot c^5 \cdot m) + \sum_{v \in V} O(4^c \cdot c^3 \cdot \deg(v)) = O(4^c \cdot c^3 \cdot m). \qquad \square$$

2.2 Improved Screening of Cliques

In addition to having generalized Ito and Iwama's [14] algorithm, we now present an improved screening stage. This enables us to improve the exponential part of the overall running time from $O(4^c)$ to $O(2.89^c)$. Furthermore, the modifications facilitate parallelization of the enumeration algorithm.

First, we present a simple and efficient test for checking whether an enumerated clique is subset of a clique with a different pivot. Lemma 5 immediately follows from the clique and pivot definitions.

Lemma 5 *A c-isolated clique C with pivot v is subset of a c-isolated clique C' with pivot $u \neq v$ iff $u \in N_-(v)$ and $N(u) \supseteq C$.*

During the screening stage of the original algorithm [14], we had to compare every enumerated clique for pivot v with every maximal c-isolated clique with pivot $u \in N_-(v)$. According to Lemma 5, we can replace these comparisons with a simple test that looks for vertices in $N_-(v)$ that are adjacent to all vertices of an enumerated clique. This test takes $O(c \cdot |C|)$ time. This alone promises a conceivable speedup in practice since the number of set comparisons is significantly reduced. Furthermore, since the enumerations of cliques for different pivots now run completely independent from each other, we can parallelize our algorithm by executing the pivot procedures for different pivot vertices on up to n different processors.

Second, we can show that we do not have to perform any brute-force set comparisons at all. Suppose an enumerated c-isolated clique C with pivot v is not maximal. Then there must be a vertex set S such that $C \cup S$ is a c-isolated clique. Obviously, S must be a subset of $N[v] \setminus C$. Also, S must be a clique and all vertices in S have to be adjacent to all vertices in C. Let D be the subset of $N[v] \setminus C$ that contains exactly the vertices that are adjacent to all vertices in C. To test the maximality of C, we first enumerate all maximal cliques $D' \subseteq D$. As a consequence, for each such clique D', the set $C \cup D'$ is a clique. If $C \cup D'$ is also c-isolated, then C is clearly not maximal and thus removed from the output. If $C \cup D'$ is not c-isolated however, then we have to check whether there is a c-isolated subset of $C \cup D'$ that is also a superset of C. This can be done by removing the vertices of highest degree from D' until either $C \cup D'$ becomes c-isolated or D' is empty. In the first case, C is

not a maximal c-isolated clique and is thus removed from the output. In the second case, C is a maximal c-isolated clique in $C \cup D'$. If this can be shown for all maximal cliques $D' \in D$, then C is a maximal c-isolated clique in G. With this maximality test, we can improve the asymptotic running time bound of the enumeration algorithm.

Theorem 2 *All maximal c-isolated cliques of an m-edge graph can be enumerated in $O(2.89^c \cdot c^2 \cdot m)$ time.*

PROOF. Since the trimming stage and enumeration stage of the algorithm have not changed, their overall running time amounts to

$$O(c^3 \cdot m + 2^c \cdot c^5 m) = O(2^c \cdot c^5 \cdot m),$$

as shown in the proof of Proposition 1. In the screening stage of the pivot procedure, we have to test each clique for maximality. At most $2^{c-1} \cdot c$ cliques are enumerated during the enumeration stage of the pivot procedure for a pivot v. For any enumerated c-isolated clique C, we have to enumerate all maximal cliques in a subset of $N[v] \setminus C$. Since $|N[v] \setminus C| \leq c - 1$, this enumeration can be performed in $O(3^{c/3})$ time [26]. For each pair of an enumerated c-isolated clique C and a maximal clique D' we decide whether a subset of $C \cup D'$ is c-isolated by successively removing the vertices with highest degree from D'. Clearly, this can be done in $O(c)$ time. Altogether, one execution of the screening stage thus has a worst-case running time of

$$O(2^c \cdot c) \cdot O(3^{c/3}) \cdot O(c) = O(2.89^c \cdot c^2).$$

There are n runs of the screening stage and together with the running times of the other stages we achieve a total worst-case running time of

$$O(2^c \cdot c^5 \cdot m) + O(2.89^c \cdot c^2 \cdot n) = O(2.89^c \cdot c^2 \cdot m). \qquad \square$$

Note that since we do not perform any pairwise comparisons anymore, we have to ensure that no clique is output twice. We can deal with this by choosing an appropriate data structure, for example a hash table, for output.

3 New Isolation Concepts

Since isolation is not merely a means of developing efficient algorithms for the enumeration of cliques but also a trait in its own right, it makes sense to consider varying degrees of isolation. For instance, this is useful for the

enumeration of isolated dense subgraphs for the identification of communities, which play a strong role in the analysis of biological and social networks [23].

The definition of c-isolation is not particularly tailored to these applications and we propose two alternative isolation concepts. One of them, min-c-isolation, is a weaker notion than c-isolation and the other, max-c-isolation, is a stronger notion than c-isolation. For both isolation concepts, we achieve a considerable speedup in the exponential part of the running time.

3.1 Minimum Isolation

Min-c-isolation is a weaker concept of isolation than the previously defined c-isolation, since we only demand that a set contains at least one vertex with less than c outgoing edges.

Definition 2 *Let $G = (V, E)$ be a graph. A vertex set $S \subseteq V$ is* min-c-isolated *if there is at least one vertex in S with less than c neighbors in $V \setminus S$.*

Obviously, every c-isolated set is also min-c-isolated (also compare Figure 1). The enumeration of maximal min-c-isolated cliques consequently yields sets that are at least as large and often larger than c-isolated cliques.

The algorithm for the enumeration of maximal min-c-isolated cliques is mainly a simplification of the algorithm from Section 2. By way of contrast to c-isolated cliques (see Theorem 2), we lose linear-time solvability for constant c; the running time now becomes $O(n \cdot m)$. We use the same pivot definition and enumerate cliques for each possible pivot; from our definition of min-c-isolation it follows directly that the pivot of a min-c-isolated clique must have less than c neighbors outside of the clique. Subsequently, we point out the differences in the three main stages of the pivot procedure.

Trimming stage. For pivot v, we again start with $C := N_+[v] := \{u \in N[v] \mid u > v\} \cup \{v\}$ as candidate set. In contrast to the trimming stage for c-isolation, we establish only one invariant for each $u \in C$:

(a) u has at least $|C| - c$ adjacent vertices in C.

The correctness of the invariant is easy to see. A vertex u with less than $|C| - c$ adjacent vertices in C cannot be in a min-c-isolated clique with pivot v since v has at least c neighbors outside any clique that contains both v and u.

Since the min-c-isolation concept makes no assumption about vertices other than the pivot, we cannot remove vertices because of their degree. This results

in the loss of the linear running time for constant c.

Enumeration Stage. Let $\tilde{c}$ be the number of vertices that can still be removed from the candidate set C after the trimming stage. We first build the complement graph $\overline{G[C]}$. Then, we enumerate the minimal vertex covers of size at most $\tilde{c}$ in $\overline{G[C]}$. We ensure that no vertex cover is enumerated twice by storing them in an appropriate data structure. From each enumerated minimal vertex cover, we directly obtain a min c-isolated clique. This follows from the fact that we have removed less than c neighbors of v during the trimming and enumeration stage. Consequently, v has less than c neighbors outside of the clique, which exactly fits our definition of min-c-isolation.

Screening Stage. In the screening stage, we remove enumerated cliques that are not maximal. The maximality of an enumerated min-c-isolated clique C can be easily tested. We simply check whether there is a vertex that is adjacent to all vertices in C. This is sufficient because any maximal min-c-isolated clique is also a maximal clique: adding a vertex that is adjacent to all vertices of a min-c-isolated clique always produces a min-c-isolated clique. Furthermore, no vertex that was removed in order to establish invariant (a) or during the enumeration stage is adjacent to all vertices in C. Hence, we only need to consider vertices in $N[v] \setminus N_+[v] = N_-(v)$. Compared to c-isolation, the fact that we only have to perform this very simple maximality test results in an improved exponential part of the running time.

Theorem 3 *All maximal min-c-isolated cliques of an n-vertex and m-edge graph $G = (V, E)$ can be enumerated in $O(2^c \cdot c \cdot m + n \cdot m)$ time.*

PROOF. The correctness of the presented enumeration algorithm follows directly from the description of the three stages of the algorithm. We complete the proof by bounding the running time of the algorithm. We call the pivot procedure for each vertex $v \in V$. During the pivot procedure for pivot v we spend $O(m)$ time in the trimming stage, $O(m + c \cdot \deg(v))$ time for the construction of $\overline{G[C]}$, and $O(2^c \cdot c^2)$ time for the enumeration of minimal vertex covers. Since at most 2^c minimal vertex covers are enumerated, we construct at most 2^c cliques in the enumeration stage. For each of these cliques, we then check whether there is a vertex in $N_-(v)$ that is adjacent to all vertices of the clique. Note that $|N_-(v)| < c$, since otherwise there is no min-c-isolated with pivot v. Consequently, we can perform the maximality test in $O(c \cdot \deg(v))$ time for each clique and thus in $O(2^c \cdot c \cdot \deg(v))$ time for all enumerated cliques. In total, the n executions of the pivot procedure have a running time

of

$$\sum_{v \in V} O(m + c \cdot \deg(v) + 2^c \cdot c^2 + 2^c \cdot \deg(v)) = O(2^c \cdot c \cdot m + n \cdot m). \qquad \square$$

3.2 Maximum Isolation

Compared to c-isolation, max-c-isolation is a stronger notion (see also Figure 1). In particular, we limit the number of outgoing edges for all vertices of the clique.

Definition 3 *Let $G = (V, E)$ be a graph. A vertex set $S \subseteq V$ is* max-c-isolated *if every vertex $v \in S$ has less than c neighbors in $V \setminus S$.*

This isolation concept is useful when we particularly want to exclude high-degree vertices from the enumerated sets. Compared to the enumeration of min-c-isolated cliques or c-isolated cliques, this can result in the enumeration of smaller cliques for equal values of c.

We apply the algorithm scheme presented in Section 2, that is, for every vertex $v \in V$ we enumerate all maximal max-c-isolated cliques with pivot v.

Trimming Stage. We compute a candidate set $C \subseteq N_+[v]$ by removing every vertex from $N_+[v]$ that cannot be in a max-c-isolated clique with pivot v. After that, each vertex $u \in C$ complies with the following invariants:

(a) $\deg(u) < |C| + c - 1$ and
(b) u has at least $|C| - c$ adjacent vertices in C.

Both invariants can be demanded because every vertex in a max-c-isolated clique must have less than c neighbors outside of the clique and all cliques with pivot v are subsets of C. Invariant (a) stems from the observation that vertices with at least c neighbors outside of C cannot be in a max-c-isolated clique that is a subset of C. Invariant (b) was also part of the trimming stages for the enumeration of c-isolated and min-c-isolated cliques. Clearly, if the invariant was correct for the weaker isolation concepts, then it must be also correct for the strongest isolation concept.

The purpose of invariant (a) is to remove vertices of high degree without scanning their adjacency lists. As a result, we can achieve a linear running time bound for constant c as was also the case for c-isolation.

Enumeration Stage. We enumerate max-c-isolated cliques $C' \subseteq C$ with pivot v. As in Section 2, we first enumerate maximal cliques in C via enumeration of minimal vertex covers of size at most $\tilde{c}$ in $\overline{G[C]}$, where $\tilde{c}$ is the number of vertices that can still be removed from the candidate set C after having possibly removed vertices in the trimming stage.

The cliques thus obtained may violate the isolation condition, since they may contain vertices with too many outgoing edges. We can restore the isolation condition for each enumerated clique by simply removing these vertices. This is done until either the resulting clique is max-c-isolated or we have removed more than $\tilde{c}$ vertices. In the latter case we discard the clique. The remaining enumerated cliques are not necessarily maximal, and therefore non-maximal cliques must be removed from the output in the screening stage.

Screening Stage. There are two possibilities for an enumerated clique C to be non-maximal. First, it can be a proper subset of another max-c-isolated clique with pivot v. Second, it can be a proper subset of a max-c-isolated clique with pivot $u < v$. For the first possibility, we test whether there is a set of vertices $D \subseteq N_+[v] \setminus C$ such that $C \cup D$ is a max-c-isolated clique. Clearly, D has to form a clique and all its vertices have to be adjacent to all vertices in C. Furthermore, whenever D contains a vertex w with degree $|C|+c+x$, then $|D|$ must have size at least $x+1$. Otherwise, $C \cup D$ is not max-c-isolated, because u has at least c outgoing edges from $C \cup D$. Hence, we test for all $0 \leq x < c-1$ whether the set

$$D^x := \{w \in N_+[v] \setminus C \mid \ C \subseteq N(w) \wedge \deg(w) \leq |C| + c + x\}$$

contains a clique of size at least $x+1$. If this is not the case for any x, then C is a maximal max-c-isolated clique for pivot v. Otherwise, C is removed from the output. Note that since $D^x \subseteq N_+[v] \setminus C$ it has size at most $c-1$: otherwise C would not be max-c-isolated.

It remains to check whether C is a proper subset of a clique with another pivot $u < v$. This can be tested in the manner described in Section 2.2, since Lemma 5 also applies to max-c-isolation. The running time of the pivot procedure is dominated by the first maximality test of the screening stage. For each of the $O(2^c)$ enumerated cliques, we have to solve Maximum Clique up to c times. Since $|D^x| < c$ for all $0 \leq x < c-1$, this can be done in $O(1.22^c)$ time [24]. [5] The overall running time of this test is then

$$O(2^c \cdot 1.22^c \cdot c) = O(2.44^c \cdot c).$$

[5] Note that this algorithm requires exponential space. A simple algorithm that runs in polynomial space and is only marginally slower was given by Fomin et al. [9].

The running time of the whole enumeration can be bounded in a similar way as in Section 2.1; we omit the details.

Theorem 4 *All maximal max-c-isolated cliques of an m-edge graph can be enumerated in $O(2.44^c \cdot c \cdot m)$ time.*

4 Isolated s-Plexes

In many applications such as social [25] and biological [4] network analysis, cliques have been criticized for their overly restrictive nature or modelling disadvantages. Hence, more relaxed concepts of dense subgraphs such as s-plexes [25,18] are of interest. An s-plex is a degree-based relaxation of the clique concept:

Definition 4 *Let $G = (V, E)$ be a graph. A subset of vertices $S \subseteq V$ of size k is called an* s-plex *if the minimum degree in $G[S]$ is at least $k - s$.*

Cliques are nothing but 1-plexes. The parameterized problem of finding an s-plex of maximum cardinality can be stated as follows:

MAXIMUM s-PLEX
Input: A graph $G = (V, E)$ and nonnegative integers s and k.
Question: Is there an s-plex $S \subseteq V$ of size at least k?

The NP-completeness of MAXIMUM s-PLEX follows from a general result on vertex deletion problems [19], but was also shown by reduction from MAXIMUM CLIQUE [1,18]. First, we strengthen the result by the corresponding parameterized hardness result (Section 4.1). Second, we indicate how to enumerate isolated s-plexes (Section 4.2).

4.1 Maximum s-Plex is W[1]-Hard

Since the previous reductions from MAXIMUM CLIQUE [1,18] transform a clique of size k into an s-plex of size $k + (s-1) \cdot n$, they are not parameterized reductions and thus do not yield any information about the parameterized complexity of the problem. We present a simple parameterized reduction from MAXIMUM CLIQUE to MAXIMUM s-PLEX for arbitrary s and thus show that the problem is W[1]-hard with respect to the combined parameter (k, s). In the following, we describe the transformation of a MAXIMUM CLIQUE instance into a MAXIMUM s-PLEX instance. An example of this transformation is given in Figure 4. Given an instance $(G = (V, E), k)$ of CLIQUE, we construct a

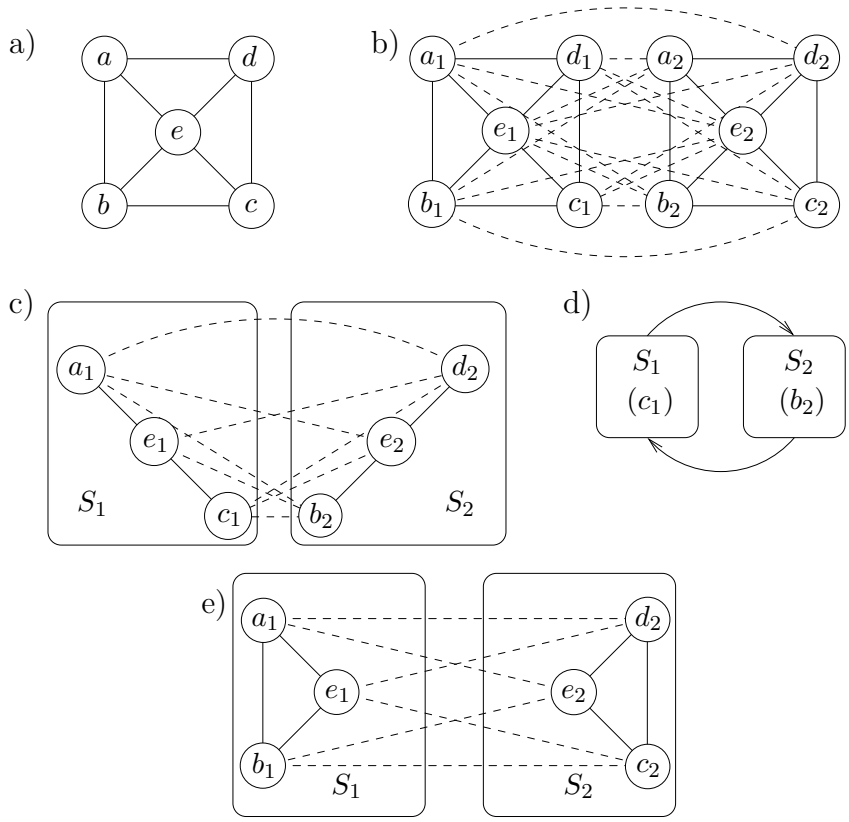

Fig. 4. Example for the reduction of Maximum Clique to Maximum 2-plex. a) Original graph G with four cliques of size $k = 3$. b) Transformed graph G' consisting of $s = 2$ compartments. Solid lines represent edges inside compartments, dashed lines those between two compartments. c) 2-plex of size $6 = s \cdot k$ with none of the compartments forming a clique of size $k = 3$. d) After arbitrarily picking vertices c_1 and b_2 the resulting exchange graph has a cycle of length two. e) After the exchange of the two vertices both compartments form cliques of size $k = 3$.

graph $G' = (V', E')$ in the following way: V' is comprised of $s \cdot |V|$ vertices, with each $v \in V$ corresponding to exactly s vertices in V':

$$V' := \bigcup_{i=1}^{s} V_i \quad \text{where} \quad V_i = \{r_i : 1 \leq r \leq n\}.$$

Each V_i is called a *compartment* of V'. Two vertices in the new graph are adjacent when the vertices they correspond to were adjacent in the original graph:

$$E' := \{\{r_i, t_j\} : i, j \leq s \wedge \{r, t\} \in E\}.$$

In other words, to generate G', every vertex in G is replaced by s vertices forming an independent set but otherwise preserving the connection structure. To show that this is indeed a parameterized reduction, one can prove the following:

Lemma 6 *G has a clique of size k iff G' has an s-plex of size $s \cdot k$.*

PROOF. Clearly, if G has a clique of size k, then G' has an s-plex of size $s \cdot k$. It remains to be shown that if G' has an s-plex S with $|S| = s \cdot k$, then G has a clique of size k. We call two vertices r_i and r_j, that is, two vertices that correspond to the same vertex r in the original graph, *siblings*. Note that two siblings are nonadjacent and have the same neighborhood: $N(r_i) = N(r_j)$. Analogously to the compartments of V', the sets $S_i := V_i \cap S$ are called the compartments of S. When S is an s-plex of size $s \cdot k$, we can suppose that S is made up of exactly s compartments each of size exactly k. Otherwise, we can always remove one vertex r_i that belongs to a compartment S_i of size more than k and insert one of its siblings r_j into a compartment S_j that has less than k vertices. This operation does not change the degree of any of the vertices in S, since two siblings have the same neighborhood. It can also be repeated as long as there is one compartment of size more than k. As a consequence, we can assume that S is made up of exactly s compartments of size exactly k.

We face two cases: none of the compartments of S forms a clique or at least one does so. Consider the latter case with S_i being a compartment of S that forms a clique: obviously the set $C = \{r \mid r_i \in S_i\}$ forms a clique of size k. We complete the proof by showing that whenever we face the first case, then we can transform S such that one of its compartments forms a clique of size k.

Since none of the compartments forms a clique of size k, there is at least one vertex r_i in every compartment S_i that has less than $k-1$ neighbors in its compartment and exactly k neighbors in at least one other compartment, because it has to have at least $s \cdot k - s = s \cdot (k-1)$ neighbors in S.

If we arbitrarily pick a vertex with this property for each compartment, we can build a directed *exchange graph* with the compartments as vertices and a directed edge from compartment S_i to compartment S_j when the vertex picked in S_i has k neighbors in S_j. An example of two compartments and the exchange graph defined by them is shown in Figure 4. The exchange graph has at least one cycle, because every vertex in it has at least one edge with outgoing direction. Along this cycle, we can exchange the vertices in the direction of the edges, meaning that we remove vertex r_i from S_i and insert its sibling r_j into S_j when S_i and S_j belong to the cycle and there is a directed edge from S_i to S_j. Note that r_j was not already part of S_j, because r_i had k neighbors in S_j, and S_j has size exactly k.

After executing the exchange for each edge of the cycle, every compartment holds again k vertices and its average degree has increased, because in each compartment on the cycle we have removed a vertex with less than $k-1$

neighbors and inserted a vertex with k neighbors—which means that it has $k-1$ neighbors after the removal of some other vertex from the compartment.

This procedure can be repeated as long as none of the compartments forms a size-k clique, and since it continuously increases the average degree of some compartments, one of them has to become a clique at some point, meaning that there also has to be a clique of size k in G. □

Theorem 5 MAXIMUM s-PLEX *is W[1]-hard with respect to the combined parameter* (s, k).

PROOF. Clearly, the described reduction can be performed in polynomial time. Furthermore, it is a parameterized reduction, as shown in Lemma 6. Since MAXIMUM CLIQUE is W[1]-hard [6] with respect to the parameter "solution size", MAXIMUM s-PLEX is also W[1]-hard. □

Theorem 5 proves that there exist no fixed-parameter algorithms for MAXIMUM s-PLEX with respect to the combined parameter (s, k), unless there is a structural collapse between FPT and the W-hierarchy. This W[1]-hardness result also holds when we parameterize the problem with either one of s and k. As a consequence, we need to consider other parameters to efficiently find maximum s-plexes or to enumerate all maximal s-plexes. This again leads to considering isolation as a parameter, as we do next.

4.2 Enumeration of Isolated s-Plexes

We present an algorithm for the enumeration of maximal min-c-isolated s-plexes that runs in FPT time with respect to parameter c for any constant s. Although it is conceivable that c-isolation or max-c-isolation can be similarly applied to the enumeration of s-plexes, both of these isolation concepts are more complicated than min-c-isolation and would hence lead to rather involved enumeration algorithms. Since it is our aim to show that the parameter of isolation in general can be applied to s-plex enumeration, we subsequently focus on the pairing of s-plexes and min-c-isolation.

Compared to the enumeration of maximal min-c-isolated cliques, we face two obstacles when enumerating maximal min-c-isolated s-plexes.

The first obstacle is that we cannot use the algorithm for the enumeration of minimal vertex covers, since an s-plex does not necessarily induce an independent set in the complement graph. The second obstacle is that an s-plex with

a pivot vertex v is not necessarily a subset of $N[v]$. We begin with describing how to overcome the first obstacle.

Since in an s-plex S of size k every vertex $v \in S$ is adjacent to at least $k - s$ vertices, the subgraph induced by S in the complement graph $\overline{G[S]}$ is a graph with maximum degree at most $s - 1$. Consider therefore the following generalization of a vertex cover:

Definition 5 *Let $G = (V, E)$ be a graph. We call a subset of vertices $S \subseteq V$ a* bounded-degree-deletion-d set *(bdd-d set) if $G \setminus S$ has maximum degree at most d.*

Clearly, a vertex cover is the same as a bdd-0 set. The idea is to enumerate maximal s-plexes in G by enumerating minimal bdd-d deletion sets with $d := s - 1$ in the complement graph $\bar{G}$. We present a fixed-parameter algorithm for the enumeration of minimal bdd-d sets that uses the size of the solution sets as the parameter. The enumeration version of the problem can be stated as follows:

Bounded-Degree-Deletion
Input: A graph $G = (V, E)$ and nonnegative integers k and d.
Task: Find all minimal bdd-d sets of size at most k of G.

The decision version of Bounded-Degree-Deletion was also considered by Nishimura et al. [22] who presented an $O((d + k)^{k+3} \cdot k + n(d + k))$ time algorithm.[6] In the following theorem, we improve upon this running time while also covering the enumeration version. The simple idea is to pick a vertex v with more than d neighbors, and then to branch into $d + 2$ cases corresponding to the deletion of v or the deletion of one vertex of an arbitrary set of $d + 1$ neighbors of v:

Theorem 6 *Let G be an n-vertex and m-edge graph G and k a nonnegative integer. There are at most $(d + 2)^k$ minimal bdd-d sets of size at most k, and they can be enumerated in $O((d + 2)^k \cdot (k + d)^2 + n \cdot (k + d))$ time.*

PROOF. First, we perform a data reduction that simply removes all vertices with degree greater than $k + d$. This is correct because every vertex v with more than $k+d$ neighbors must belong to any bdd-d set. Finding these vertices can be performed in $O(n \cdot (k + d))$ time by scanning the adjacency list of each vertex up to position $k + d$. Furthermore, since we remove at most k vertices

[6] It has been recently shown that the decision version also admits a so-called problem kernel consisting of at most $(d^3+4d^2+6d+4) \cdot k$ vertices [7]. Moreover, there are first experimental results concerning the efficient finding of maximum-cardinality s-plexes [20].

from the graph, this removal can be done in $O(n \cdot k)$ time. Any of the deleted vertices must be later on added to the enumerated minimal bdd-d sets.

After having performed the data reduction, a search tree procedure enumerates all minimal bdd-d sets of the graph. The main idea is to choose a vertex v that has more than d neighbors as long as such a vertex exists, and then branch on the set $N_{d+1}[v]$ that contains v and $d+1$ of its neighbors. Obviously, at least one of the vertices in $N_{d+1}[v]$ must belong to any bdd-d set, since otherwise vertex v would retain more than d neighbors. Consequently, we branch into $d+2$ cases and enumerate the minimal bdd-d sets of $(G-u, k-1)$ for each $u \in N_{d+1}[v]$. Also, for each branch we keep track of the set of vertices S that has been deleted so far. The branching is performed as long as $k \geq 0$ and a vertex v with more than d neighbors can be found.

If at a search tree node $k < 0$, then the set of removed vertices is already too large and therefore we stop branching. Otherwise, if the graph already has maximum degree d and $k \geq 0$, then the set of removed vertices S is a bdd-d set. We check whether S is minimal, that is, whether there is no vertex $u \in S$ such that $S \setminus \{u\}$ is a bdd-d set, and output all minimal S. We add the vertices that have been removed during the data reduction stage and obtain a minimal bdd-d set of the input graph.

Based on the above description, it is not hard to prove the claimed running time bound; we omit the basically straightforward details. □

The search tree algorithm could possibly output duplications of a solution set. Again we cope with this by outputting the solutions in an appropriate data structure. The exponential part of the running time cannot be improved if the goal is to output all minimal bdd-d sets: the graph that consists of k disjoint cliques of size $d+2$ has exactly $(d+2)^k$ minimal bdd-d sets.

Now we turn to the second obstacle in the enumeration of maximal min-c-isolated s-plexes. Recall that the second obstacle lies in the fact that given a pivot vertex v, maximal min-c-isolated s-plexes with pivot v are not necessarily a subset of $N_+[v]$, since they can contain up to $s-1$ vertices that are not adjacent to v. We deal with this by enumerating all maximal min-c-isolated s-plexes for a given *pivot set* instead of a single pivot. The *pivot set* of a min-c-isolated s-plex is defined as the set that contains the pivot vertex v of the s-plex and those vertices that belong to the s-plex but are not adjacent to v. The *pivot vertex* is defined as the vertex with lowest index among the vertices with less than c outgoing edges. There has to be at least one such vertex, since otherwise the condition of min-c-isolation would be violated, but it does not necessarily have to be the vertex with the lowest index of all vertices in the s-plex. Consider, for example, a min-1-isolated 3-plex S of size k that contains a

	procedure pivot-min-isolated-s-plex(G, v, c)
Input:	A graph $G = (V, E)$ with vertices sorted by degree, a vertex $v \in V$ and a nonnegative integer c.
Output:	The set $\mathcal{C}$ of maximal min-c-isolated s-plexes with pivot v.
	Trimming stage
1:	$C := N(v)$
2:	$\tilde{c} := c - 1$
3:	**foreach** $u \in C$
4:	**if** $\lvert N(u) \cap C \rvert \leq \lvert N(v) \rvert - c - s$
5:	$C := C \setminus \{u\}$; $\tilde{c} := \tilde{c} - 1$
6:	**if** $\tilde{c} < 0$ **return** $\emptyset$
	Enumeration stage
7:	construct $\overline{G[C]}$
8:	**foreach** $P \in \{P' \subseteq V \setminus N[v] \mid \lvert P' \rvert \leq s - 1\}$
9:	$P := P \cup \{v\}$
10:	construct $\overline{G[C \cup P]}$
11:	enumerate all minimal bdd-$(s-1)$ sets $D \subseteq C$ of $\overline{G[C \cup P]}$ of size at most $\tilde{c}$
12:	$\mathcal{C} := \mathcal{C} \cup \{(C \cup P) \setminus D \mid D$ is obtained in line 11$\}$
	Screening stage
13:	**foreach** $S \in \mathcal{C}$
14:	**if** $\exists u \in S :\ u < v$ and u has less than c neighbors in $V \setminus S$
15:	remove S from $\mathcal{C}$
16:	**foreach** $S \in \{S' \in \mathcal{C} \mid S'$ has pivot set of size at most $s - 1\}$
17:	**if** $\exists u \in V \setminus S :\ \{u\} \cup S$ is an s-plex
18:	remove S from $\mathcal{C}$
19:	**return** $\mathcal{C}$

Fig. 5. The pivot procedure for enumerating maximal min-c-isolated s-plexes.

vertex u that has $k-3$ neighbors in S and one outgoing edge and a vertex v that has $k-1$ neighbors in S and no outgoing edge. Since $\deg(u) < \deg(v)$, $u < v$. However, u cannot be pivot since it has one neighbor in $V \setminus S$.

The overall structure of the algorithm remains the same, that is, for each $v \in V$, we enumerate all maximal min-c-isolated s-plexes with pivot v. Below, we describe the pivot procedure for pivot v. The procedure also consists of the three stages, and the pseudo code is given in Figure 5.

Trimming stage. We start with $C := N(v)$ as candidate set. In lines 2–5 of Figure 5, we remove vertices from C that have too few neighbors in C. After this removal, the following invariant holds for all $u \in C$:

(a) u has at least $\lvert N(v) \rvert - c - s + 1$ neighbors in C.

This invariant must be fulfilled since any s-plex S that contains v and a vertex u with at most $|N(v)| - c - s$ neighbors has size at most $|N(v)| - c$. Hence v has at least c neighbors in $V \setminus S$ and is not the pivot of S.

Enumeration stage. We first build the complement graph $\overline{G[C]}$ in line 7 of the algorithm. Then, for each possible pivot set P with pivot v, we independently enumerate the maximal min-c-isolated s-plexes. This is done by first extending the complement graph to $\overline{G[C \cup P]}$ (line 10) and then enumerating minimal bdd-$(s-1)$ sets of size at most $c-1$ in $\overline{G[C \cup P]}$ (line 11). Note that we also explicitly demand that the enumerated bdd-$(s-1)$ sets do not contain any vertices from P, since we want to enumerate s-plexes with pivot set P. In other words, we demand that the enumerated bdd-$(s-1)$ sets are a subset of C. We can do this by solving an annotated version of Bounded-Degree-Deletion. From each enumerated bdd-$(s-1)$ set D, we then obtain an s-plex by removing D from $C \cup P$ (line 12). Note that the enumerated s-plexes are min-c-isolated because we have removed less than c vertices from C in the trimming and enumeration stage.

Screening stage. In the screening stage, we remove enumerated s-plexes that either have pivot $u \neq v$ or are not maximal. First, we test whether S contains a vertex $u < v$, where u has less than c neighbors outside of the s-plex (lines 13–15). Then S has pivot u and not v, and we remove S from the output. Next, we test whether there is a vertex u such that $S \cup \{u\}$ is an s-plex (lines 16–18). Then S is not a maximal min-c-isolated s-plex, and hence it is removed from the output. Note that any enumerated min-c-isolated s-plex S with pivot set P cannot be subset of a min-c-isolated s-plex S' with pivot set P because we have only enumerated minimal bdd-$(s-1)$ sets. For the same reason, S can also not be subset of a min-c-isolated s-plex S' with pivot set $P' \subset P$. Hence, S can only be subset of a min-c-isolated s-plex S' that has a pivot set $P' \supset P$. Therefore, we only need to perform the test on enumerated s-plexes that have a pivot set of size at most $s-1$.

Theorem 7 *All maximal min-c-isolated s-plexes of a graph can be enumerated in $O((s+1)^c \cdot (s+c) \cdot n^{s+1} + n \cdot m)$ time.*

PROOF. The correctness of the algorithm follows from the description above. We complete the proof by bounding the running time of the algorithm. The trimming stage of the pivot procedure can be performed in $O(m)$ time, since each edge is visited a constant number of times. The construction of $\overline{G[C]}$ can be performed in $O(m + (s+c) \cdot n)$ time. This is because C contains at most m edges and at most $(s+c) \cdot n$ non-edges after the trimming stage. Hence, $|C|^2 \leq m + (s+c) \cdot n$. Next, we enumerate minimal bdd-$(s-1)$ sets

of size at most $c-1$ for all possible pivot sets P for pivot vertex v. For each pivot set P, we extend the complement graph so that it also contains P. This can be done in $O(s \cdot n)$ time, because P contains at most s vertices. According to Theorem 6, the enumeration of the minimal bdd-$(s-1)$ sets can be performed in $O((s+1)^c \cdot (s+c)^2 + n \cdot (s+c))$ time. Furthermore, we can omit the data reduction of the algorithm for the enumeration of minimal bdd-$(s-1)$ sets, since after the trimming stage, every vertex from C has at most $s+c$ neighbors in $\overline{G[C]}$. Therefore, we save the $O(n \cdot (s+c))$ part of this running time. As shown by Theorem 6, at most $(s+1)^c$ bdd-d sets are enumerated. Consequently, the construction of s-plexes for pivot set P in line 12 can be performed in time $O((s+1)^c \cdot n)$. The enumeration of s-plexes for one pivot set thus takes

$$O(s \cdot n + (s+1)^c \cdot (s+c)^2 + (s+1)^c \cdot n) = O((s+1)^c \cdot (s+c) \cdot n)$$

time. During one execution of the pivot procedure, $O(n^{s-1})$ pivot sets are created and except for the construction of $\overline{G[C]}$ one must perform every step for each pivot set. Hence, the execution of the enumeration stage for one pivot vertex has a total running time of

$$O(m + (s+c) \cdot n + n^{s-1} \cdot ((s+1)^c \cdot (s+c) \cdot n)) = O((s+1)^c \cdot (s+c) \cdot n^s + m).$$

In the screening stage, one performs two tests on each enumerated s-plex. For one s-plex, the first test can be performed in $O(n)$ time when for each vertex we keep track of how many neighbors it has outside of the s-plex. Since $(s+1)^c \cdot n^{s-1}$ s-plexes are enumerated, this test has a total running time of $O((s+1)^c \cdot n^s)$.

For one s-plex, the second test can be performed in $O(n^2)$ time. Also note that it has to be performed only for s-plexes that have pivot sets of size at most $s-1$. There are $O(n^{s-2})$ of these pivot sets and at most $(s+1)^c$ s-plexes have been enumerated for each of these pivot sets. Consequently, this test can be performed in $O((s+1)^c \cdot n^s)$ time. The screening stage thus has a total running time of $O((s+1)^c \cdot n^s)$. In total, one execution of the pivot procedure has a running time of $O((s+1)^c \cdot (s+c) \cdot n^s + m)$. The pivot procedure is executed n times, once for each vertex $v \in V$. This results in a total running time of

$$n \cdot O((s+1)^c \cdot (s+c) \cdot n^s + m) = O((s+1)^c \cdot (s+c) \cdot n^{s+1} + n \cdot m). \qquad \square$$

By Theorem 7, for every constant s, we obtain a fixed-parameter algorithm for enumerating all maximal min-c-isolated s-plexes with respect to the parameter c. Note, however, that the algorithm is not a fixed-parameter algorithm with respect to the combined parameter (s, c) and that such an algorithm does not exist because there can be too many maximal min-c-isolated s-plexes in a

graph. Consider, for example, a graph that consists of two cliques, a clique C_1 of size $s-1$ and a clique C_2 of size $n-s+1$. The union of C_1 and any subset of C_2 of size $s-1$ is a maximal min-1-isolated s-plex of the graph. Hence, this graph has at least $\binom{n}{s-1}$ maximal min-1-isolated s-plexes. Therefore, their enumeration cannot be done in $f(s,c) \cdot n^{O(1)}$ time.

5 Outlook

There are numerous avenues for future research. First, we studied the enumeration of s-plexes only with respect to min-c-isolation, leaving open the cases of c-isolation and max-c-isolation. Especially interesting seems to be the question whether one can achieve fixed-parameter tractability results with respect to the parameter (s,c) for a combination of s-plexes with one of these two isolation concepts. Second, it is open to derive a *linear-time* algorithm for enumerating all maximal min-c-isolated cliques for constant c. Third, it is conceivable that we could speed up our enumeration algorithms by using compact representations for the output maximal cliques, as already proposed for the enumeration of minimal vertex covers [5]. Fourth, also further clique relaxations—for example the paraclique concept [4]—might be studied using the isolation parameter.

Finally, there is new empirical evidence for the usefulness isolated clique enumeration in various applications. We have implemented and tested our algorithms based on algorithm engineering methods [12], demonstrating that the proposed algorithms are fast not only in theory, but also in practice.[7] Furthermore, we showed that in financial networks, isolated cliques have interesting properties compared to arbitrary maximal cliques [12]. We conclude that the enumeration of isolated cliques should also be studied in other real-world networks.

Acknowledgments We thank an anonymous referee of *Theoretical Computer Science* for valuable comments that have improved the presentation of this paper, Hiro Ito and Kazuo Iwama (Kyoto) for making the manuscript of their journal version [14] available to us, and Jiong Guo (Jena) for the idea for Theorem 6.

[7] The corresponding program is free software and available from `http://theinf1.informatik.uni-jena.de/c-isol/`.

References

[1] B. Balasundaram, S. Butenko, I. V. Hicks, and S. Sachdeva. Clique relaxations in social network analysis: The maximum k-plex problem, 2008. Manuscript.

[2] B. Balasundaram, S. Butenko, and S. Trukhanovzu. Novel approaches for analyzing biological networks. *Journal of Combinatorial Optimization*, 10(1):23–39, 2005.

[3] S. Butenko and W. E. Wilhelm. Clique-detection models in computational biochemistry and genomics. *European Journal of Operational Research*, 173(1):1–17, 2006.

[4] E. J. Chesler, L. Lu, S. Shou, Y. Qu, J. Gu, J. Wang, H. C. Hsu, J. D. Mountz, N. E. Baldwin, M. A. Langston, D. W. Threadgill, K. F. Manly, and R. W. Williams. Complex trait analysis of gene expression uncovers polygenic and pleiotropic networks that modulate nervous system function. *Nature Genetics*, 37(3):233–242, 2005.

[5] P. Damaschke. Parameterized enumeration, transversals, and imperfect phylogeny reconstruction. *Theoretical Computer Science*, 351(3):337–350, 2006.

[6] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.

[7] M. R. Fellows, J. Guo, H. Moser, and R. Niedermeier. A generalization of Nemhauser and Trotter's local optimization theorem. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science (STACS '09)*, Dagstuhl Seminar Proceedings, pages 409–420. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2009.

[8] J. Flum and M. Grohe. *Parameterized Complexity Theory.* Springer, 2006.

[9] F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: a simple $O(2^{0.288n})$ independent set algorithm. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '06)*, pages 18–25. ACM Press, 2006.

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

[11] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[12] F. Hüffner, C. Komusiewicz, H. Moser, and R. Niedermeier. Enumerating isolated cliques in synthetic and financial networks. In *Proceedings of the 2nd Annual International Conference on Combinatorial Optimization and Applications (COCOA '08)*, volume 5165 of *LNCS*, pages 405–416. Springer, 2008.

[13] F. Hüffner, R. Niedermeier, and S. Wernicke. Fixed-parameter algorithms for graph-modeled data clustering. In S. Butenko, W. A. Chaovalitwongse, and P. M. Pardalos, editors, *Clustering Challenges in Biological Networks*. World Scientific, 2009.

[14] H. Ito and K. Iwama. Enumeration of isolated cliques and pseudo-cliques. *ACM Transactions on Algorithms*, 2008. To appear.

[15] H. Ito, K. Iwama, and T. Osumi. Linear-time enumeration of isolated cliques. In *Proceedings of the 13th European Symposium on Algorithms (ESA '05)*, volume 3669 of *LNCS*, pages 119–130. Springer, 2005.

[16] C. Komusiewicz. *Various Isolation Concepts for the Enumeration of Dense Subgraphs*. Diplomarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2007.

[17] C. Komusiewicz, F. Hüffner, H. Moser, and R. Niedermeier. Isolation concepts for enumerating dense subgraphs. In *Proceedings of the 13th International Computing and Combinatorics Conference (COCOON '07)*, volume 4598 of *LNCS*, pages 140–150. Springer, 2007.

[18] S. Kosub. Local density. In U. Brandes and T. Erlebach, editors, *Network Analysis: Methodological Foundations*, volume 3418 of *LNCS*, chapter 6, pages 112–142. Springer, 2005.

[19] J. M. Lewis and M. Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *Journal of Computer and System Sciences*, 20(2):219–230, 1980.

[20] H. Moser, R. Niedermeier, and M. Sorge. Algorithms and experiments for clique relaxations—finding maximum s-plexes. Manuscript, submitted for publication, February 2009.

[21] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[22] N. Nishimura, P. Ragde, and D. M. Thilikos. Fast fixed-parameter tractable algorithms for nontrivial generalizations of vertex cover. *Discrete Applied Mathematics*, 152(1–3):229–245, 2005.

[23] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[24] J. M. Robson. Algorithms for maximum independent sets. *Journal of Algorithms*, 7(3):425–440, 1986.

[25] S. B. Seidman and B. L. Foster. A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6(1):139–154, 1978.

[26] E. Tomita, A. Tanaka, and H. Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1):28–42, 2006.