Finding Dense Subgraphs of Sparse Graphs[⋆]

Christian Komusiewicz and Manuel Sorge[⋆⋆]

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin
{christian.komusiewicz,manuel.sorge}@tu-berlin.de

Abstract. We investigate the computational complexity of the DENSEST-k-SUBGRAPH (DkS) problem, where the input is an undirected graph $G = (V, E)$ and one wants to find a subgraph on exactly k vertices with a maximum number of edges. We extend previous work on DkS by studying its parameterized complexity. On the positive side, we show that, when fixing some constant minimum density μ of the sought subgraph, DkS becomes fixed-parameter tractable with respect to either of the parameters maximum degree and h-index of G. Furthermore, we obtain a fixed-parameter algorithm for DkS with respect to the combined parameter "degeneracy of G and $|V| - k$". On the negative side, we find that DkS is W[1]-hard with respect to the combined parameter "solution size k and degeneracy of G". We furthermore strengthen a previous hardness result for DkS [Cai, Comput. J., 2008] by showing that for every fixed μ, $0 < \mu < 1$, the problem of deciding whether G contains a subgraph of density at least μ is W[1]-hard with respect to the parameter $|V| - k$.

1 Introduction

Identifying dense regions of graphs is a fundamental computational problem with many important applications, for instance in computational biology [19] and social network analysis [3]. There are many different definitions of what a dense subgraph is [11, 17] and for almost all of these formulations, the corresponding computational problems are NP-hard.

In this work, we study the problem of finding subgraphs with a fixed number k of vertices and a maximum number of edges. This problem is known as DENSEST-k-SUBGRAPH. For fixed k, maximizing the number of edges is the same as maximizing the *density* of a graph $G = (V, E)$ which is defined as $2|E|/(|V|(|V| - 1))$. Using the notion of density, the NP-hard DENSEST-k-SUBGRAPH problem [11, 15] can be defined as follows.

DENSEST-k-SUBGRAPH (DkS) :
Input: A graph $G = (V, E)$, and a nonnegative integer k.
Task: Find a vertex set $S \subseteq V$ of size exactly k such that $G[S]$ has maximum density.

[⋆] One result of this work (Thm. 4) is contained in the first author's dissertation [16].
[⋆⋆] Supported by DFG projects PABI (NI 369/7-2) and DAPA (NI 369/12).

DkS is at least as hard as the well-studied CLIQUE problem which asks for finding a complete graph of order exactly k. In this work, our aim is to provide a better overview of when DkS becomes computationally hard or tractable, respectively. To this end, we consider how two types of parameters influence the complexity of DkS. The first type of parameters are the classical parameters "solution size k" and "parameterization by dual $|V|-k$". The second type of parameters measure the sparseness of the input graph G: maximum degree Δ, h-index,[1] and degeneracy d. Bounded maximum degree means that *all* vertices have few neighbors, bounded h-index means that *most* vertices have few neighbors, and bounded degeneracy means that *there is always* a vertex with few neighbors. By definition, $\Delta \geq h$-index $\geq d$. The study of these three parameters is motivated by two facts: First, many real-world networks such as biological and social networks are relatively sparse since they contain many vertices of low degree and only few vertices of high degree (the network "hubs"). Second, the otherwise notoriously hard CLIQUE problem is much easier on sparse graphs. For example, all maximal cliques can be enumerated in $O(d^{3/d} \cdot d \cdot n)$ time on graphs with degeneracy d [9].

We study the complexity of DkS mostly by considering the following problem which can be seen as a decision variant of DkS. Here, one asks whether there is a k-vertex subgraph with density *at least* μ, where $0 \leq \mu \leq 1$ is some fixed constant. We call such subgraphs μ-cliques, that is, a graph $G = (V, E)$ is a *μ-clique* if the density of G is at least μ.

μ-CLIQUE:
Input: A graph $G = (V, E)$, and a nonnegative integer k.
Question: Is there a vertex set $S \subseteq V$ of size at least k such that $G[S]$ is a μ-clique?

Throughout this work, we assume that μ is a fixed constant, in other words, that it is independent of k and n. This assumption can be motivated by the fact that in many applications, the dense subgraphs that one wants to find should be almost complete graphs.

Related Work. CLIQUE, which asks to find a complete subgraph of order k is W[1]-hard with respect to the parameter k and fixed-parameter tractable with respect to the dual parameter $n - k$ [7]. Finding a densest subgraph of order exactly k is NP-hard and W[1]-hard with respect to k, as it is a generalization of CLIQUE. Moreover, DkS is W[1]-hard with respect

[1] The structural graph parameter h-index was introduced by Eppstein and Spiro [8] in the context of triangle counting in dynamic graphs. For a definition, see Section 2.

Table 1: Summary of our results and previous results for μ-Clique and DkS. Note that hardness transfers from μ-Clique to DkS and tractability transfers in the reverse direction. For fixed-parameter tractability (FPT) results, we write a rough estimate of the exponential running time factor. Herein, k denotes the order of the sought μ-clique, $\ell = n - k$ is the number of vertices that have to be deleted in order to obtain a μ-clique or a densest subgraph, and d denotes the degeneracy of the input graph.

parameter	μ-Clique	DkS
max. degree Δ	FPT: $\Delta^{O(\Delta)}$ (Theorem 1), no poly. kernel (Theorem 7)	NP-hard for $\Delta = 3$ [10]
h-index h	FPT: $h^{O(h)}$ (Theorem 2), no poly. kernel (Theorem 7)	
degeneracy d	$\in$ XP (Lemma 1(iii))	NP-hard for $d = 2$ [10]
(k, d)	W[1]-hard (Theorem 6)	
ℓ	W[1]-hard (Theorem 4)	W[1]-hard [5]
(ℓ, d)		FPT: $(\ell + d)^{O(\ell)}$ (Theorem 3)

to the parameter $n - k$ [5]. It is, however, fixed-parameter tractable with respect to the combined parameter "maximum degree Δ and k" [6]. Holzapfel et al. [13] showed that DkS remains NP-hard, even when looking only for subgraphs with average degree at least $2 + \Omega(1/k^{1-\epsilon})$ for $0 < \epsilon < 2$. Finding k-vertex subgraphs of average degree at least $2 + O(1/k)$, however, can be done in polynomial time. Furthermore, DkS is NP-hard even in graphs with maximum degree three and degeneracy two [10]. The "densest subgraph" in this reduction, however, has very low, non-constant density. For an overview of computational aspects of finding dense subgraphs, we refer to the survey of Kosub [17]. A related problem is Minimum Subgraph of Minimum Degree, where the task is to find a subgraph of order at most k such that each vertex has a given minimum degree. Minimum Subgraph of Minimum Degree is W[1]-hard with respect to the parameter k but becomes fixed-parameter tractable on graphs of bounded local treewidth and graphs with excluded minors [2]. A further related problem is to find a subgraph that has maximum average degree (without constraint on the order). This problem is polynomial-time solvable using network flow techniques [12].

Our Results. Table 1 gives an overview of our results; note that all negative results that were obtained for μ-Clique immediately transfer to DkS. Our results can be summarized as follows. Finding dense subgraphs is significantly harder than finding cliques since μ-Clique and DkS are W[1]-hard with respect to the parameter (d, k). Furthermore, we show that the W[1]-hardness for DkS parameterized by $n - k$ [5] can also be generalized to hold for μ-Clique for all μ, $0 < \mu < 1$. Finally, we show that, in contrast to DkS, μ-Clique is fixed-parameter tractable for the parameters maximum degree Δ and h-index h of G. In particular, we show that the practically relevant case of finding subgraphs whose density μ deviates not too much from the maximum density (that is, $1/\mu$ is small) is still tractable for bounded Δ or h.

2 Preliminaries

We consider simple undirected graphs $G = (V, E)$ where $n := |V|$ and $m := |E|$. The *order* of a graph is the number of vertices. For a vertex set $S \subseteq V$ we denote by $N(S) := \bigcup_{v \in S} N(v) \setminus S$ the *neighborhood of* S, and by $\deg(v)$ the *degree* of v. We use $G[S]$ to denote the *subgraph induced by* S. The *degeneracy* of a graph G is the smallest integer d such that every induced subgraph of G has at least one vertex with degree at most d. The *h-index* of a graph G is the maximum integer h such that G contains h vertices of degree at least h. The property of being a μ-clique is not hereditary, but has a "nestedness" property [17]: Every μ-clique $G = (V, E)$ has an induced subgraph G' on $|V| - 1$ vertices that is also a μ-clique. For the relevant notions from parameterized complexity, refer to [7].

3 Fixed-Parameter Algorithms

Here, we present fixed-parameter algorithms for the parameters maximum degree Δ of G, h-index of G and the combined parameter that comprises $n - k$ and degeneracy of g. Before presenting these algorithms, we observe relationships between the order of μ-cliques and the sparsity parameters under consideration. We also give an observation about the enumeration of certain subgraphs in degree bounded graphs, which yields a subroutine used in our algorithms.

Preparations. The relation between the order of a μ-clique and its maximum degree, h-index and degeneracy is as follows.

Lemma 1. *A μ-clique with*
(i) maximum degree Δ has order at most $\Delta/\mu + 1$.
(ii) h-index h has order at most $\frac{h\cdot(h-1)+2\cdot(n-h)\cdot h}{\mu\cdot(n-1)} < \frac{2\cdot h}{\mu}$.
(iii) degeneracy d has order less than $(4 \cdot d + \mu)/2 \cdot \mu$.

The upper bound $\frac{h\cdot(h-1)+2\cdot(n-h)\cdot h}{\mu\cdot(n-1)}$ on the order of μ-cliques is tight as a graph consisting of a clique of order h and of $n - h$ further vertices that are an independent set but adjacent to all vertices of the clique has density exactly μ if n is equal to the upper bound.

Continuing the preparation for our tractability results, a central observation is the following.

Lemma 2. *Let G be a graph with maximum degree Δ and let v be a vertex in G. There are at most $4^k \cdot (\Delta - 1)^k$ connected subgraphs of G that contain v and have order at most k. Furthermore, these subgraphs can be enumerated in $\mathrm{O}(4^k \cdot (\Delta - 1)^k \cdot (n + m))$ time.*

Proof. We describe a search tree for enumerating these subgraphs. In each search tree node, we maintain two vertex sets P (the "pivot set") and N, where N is a subset of P and the task is to enumerate all vertex sets S such that 1) $G[S]$ is connected, 2) $P \subseteq S$, and 3) the vertices of N have no neighbors in $S \setminus P$. Furthermore, in each of the search tree nodes, there will be a distinguished *active* vertex v of $P \setminus N$. We will consider adding neighbors of the active vertex first. The details are as follows.

Assume that there is an arbitrary but fixed ordering of the vertices of G. Initialize the search by setting the pivot set $P := \{v\}$ and setting $N = \emptyset$ where v is the vertex with lowest index in the fixed ordering. Furthermore, set v as active vertex. Then, in each search tree node, do the following. First, report $G[P]$. Then, if $|P| = k$, abort this branch. Otherwise, if $|P| < k$ and there is no active vertex, then choose the vertex $v \in P \setminus N$ that has lowest index in the fixed ordering as new active vertex. Now, branch into the following cases to add neighbors of the active vertex v: First, for each neighbor u of v in $V \setminus P$ that is not adjacent to any vertex in N create a search tree branch with $(P \cup \{u\}, N)$ that is, one branch for each possibility to add a neighbor of v and keep v as active vertex in these branches. Second, create one further search tree branch with $(P, N \cup \{v\})$, that is, a branch in which we assume that no further neighbors of v may be added; in this branch v will become *inactive*.

Since a vertex never leaves P once it has been added and only neighbors of vertices in P are added to P, clearly, the graph $G[P]$ is connected, contains v and has order at most k in each search tree node. Furthermore, each connected graph that contains v and is of order at most k is equal to $G[P]$ in some search tree node: for each vertex that is a neighbor of the current pivot set and not a neighbor of N, we branch at some point into the case that this vertex is added.

To bound the number of search tree nodes, observe that at most k vertices can be added to P and, hence, at most k vertices can be added to N. Now, assume that we branch in advance into all the cases to either add a neighbor of an active vertex or move an active vertex to N, that is, we fix in advance that we add say x_1 neighbors of the first vertex, x_2 neighbors of the second active vertex and so on. The number of possible branchings is 2^{2k}, since in the first branch, we add a vertex to P and in the second branch, we add a vertex to N and the cardinality of both sets is at most k. Now, assume that we branch for each such fixed case into the different cases to add a neighbor of the active vertex v. Then, this can be done by a search tree of depth at most k and in each search tree node, we branch into at most $\Delta - 1$ cases, to add a vertex in $V \setminus P$ to P (note that except for the first branching, every active vertex v has at least one neighbor in P since $G[P]$ is connected). Hence, the overall search tree size is $\mathrm{O}(4^k \cdot (\Delta - 1)^k)$. Since the steps at each search tree node can be performed in $\mathrm{O}(n + m)$ time, the search tree also gives a $\mathrm{O}(4^k \cdot (\Delta - 1)^k \cdot (n + m))$ running time enumeration algorithm. $\square$

Next, we present fixed-parameter algorithms for the parameters maximum degree Δ and h-index. We gradually develop the algorithms starting with the (easiest) case of finding connected μ-cliques in graphs with maximum degree Δ. Then, we present an algorithm for disconnected μ-cliques in graphs with maximum degree Δ. Finally, we describe an algorithm for the parameter h-index. Note that we can restrict ourselves to finding μ-cliques of order exactly k due to the nestedness property.

Finding connected μ-cliques. We use the enumeration algorithm described in Lemma 2. For every vertex v, we start an enumeration of all connected graphs of order at most k that contain v and instead of reporting the graphs, we check whether it is a μ-clique and report it or not accordingly. Plugging in the bound for k given by Lemma 1(i), we obtain the following.

Proposition 1. *All connected μ-cliques in a graph G with maximum degree Δ can be enumerated in $\mathrm{O}(4^{\Delta/\mu+1} \cdot (\Delta - 1)^{\Delta/\mu+1} \cdot n(n + m))$ time.*

Finding disconnected μ-cliques. The idea for finding disconnected μ-cliques is to combine different connected subgraphs such that the sum of edges and vertices yields a graph with density at least μ. In the process of combining these connected μ-cliques, we have to ensure that we only combine these numbers for disjoint graphs. Otherwise, a dense subgraph might be counted twice. To this end, we use *color coding* [1] to obtain a randomized fixed-parameter algorithm with one-sided error. The algorithm can be derandomized using standard techniques with an additional running time factor of $2^{O(k)}$[1]. Assume that the input graph contains a μ-clique of order k, and let S be the vertex set of this μ-clique. The basic idea of color coding is to color the vertices of the input graph uniformly at random with a set C of k colors and to hope that S is colorful, that is, for each color in C there is exactly one vertex in S that has received this color. Assuming the graph is colored this way, first use the enumeration algorithms for connected μ-cliques described above, and then "combine" these connected graphs by applying dynamic programming. The color-coding/enumeration/dynamic programming routine is repeated sufficiently often to achieve constant error probability. The details are as follows.

After the coloring, first compute for every subset C' of C the densest connected subgraph that has color set C'. This can be easily achieved by adapting the above enumeration algorithm to only report colorful μ-cliques. Using this enumeration, we fill a table D where for each color set $C' \subseteq C$, the entry $D(C')$ contains the maximum number of edges in a connected μ-clique in G whose vertices have exactly the colors from C'. Afterwards, we find the maximum density of a colorful μ-clique of order k using another table T. Here, the entry in $T(C')$ for some color set $C' \subseteq C$ contains the number of edges of a (possibly disconnected) μ-clique with maximum density in G whose vertices have exactly the colors from C'. Observe that either $T(C') = D(C')$, or there is a partition of C' into C'_1, C'_2 such that $T(C') = T(C'_1) + T(C'_2)$. Thus, we fill $T(C')$ by the following recurrence:

$$T(C') = \max\{D(C'), \max_{C'' \subset C'} \{T(C'') + T(C' \setminus C'')\}\}.$$

The maximum density of a colorful μ-clique of order k is then found in $T(C)$. The table T can be filled in $\mathrm{O}(3^k)$ time, since there are at most this many triples (C', C'_1, C'_2) such that $C' \subset C$ and (C'_1, C'_2) partitions C' (each color in C either has to be in C'_1, C'_2, or $C \setminus C'$); for each such triple there is only one table lookup. Adding the running time for filling T, we obtain a running time of $\mathrm{O}(3^k + 4^k \cdot (\mathit{\Delta} - 1)^k \cdot n(n+m)) = \mathrm{O}(4^k \cdot (\mathit{\Delta} - 1)^k \cdot n(n+m))$, where $\mathit{\Delta}$ is the maximum degree of G.

The error probability can be bounded as follows [1]. When coloring the vertices with k colors uniformly at random, the probability of a μ-clique S being colorful is exactly $k!/k^k$, since there are k^k distinct colorings of S and $k!$ colorful ones. By Stirling's approximation, this probability is at least e^{-k} and by repeating e^k times the random coloring and the algorithm above, the probability of missing a feasible μ-clique is at most $(1 - e^{-k})^{e^k} \leq 1/e$. Using Lemma 1(i) we obtain the following.

Theorem 1. *μ-Clique can be solved in time* $O((4e \cdot (\Delta - 1))^{\Delta/\mu+1} n(n + m))$*, reporting a yes-instance as a no-instance with probability at most* $1/e$*, where* Δ *is the maximum degree in the input graph.*

It has previously been shown that, using random separation, DkS can be solved in $2^{O(\Delta k)}$ time with one-sided error and constant error probability [6]. Our algorithm above applied to DkS runs in $2^{O(\log(\Delta)k)}$ time.

Parameterization by h-index. We now describe how to adapt the algorithm from Theorem 1 to obtain a fixed-parameter algorithm for the parameter h-index of the input graph. In many practical applications the h-index is much smaller than the maximum degree. For instance, social and biological networks have few so-called hubs, that is, vertices of very high degree, and many low-degree vertices. Hence, the h-index is much smaller than the maximum degree for these graphs.

The main idea of the algorithm is as follows. Let H be the set of the at most h vertices with degree at least h, and assume that S is a vertex set of size k such that $G[S]$ is a μ-clique. First, by trying all 2^h partitions of H, guess the set H_S of vertices that are in $S \cap H$. We annotate every vertex $v \in V \setminus H$ with the number of neighbors it has in H_S. Let the *weight* of a subgraph G' of the input graph $G = (V, E)$ be the sum of the vertex annotations in G' and the number of edges in G'. Now the task is to find a subgraph in $G[V \setminus H]$ of order at most $k - |H_S|$ that has maximum weight. If we have such subgraphs for all possible choices of H_S, we can compare them, also accounting for the edges in $G[H_S]$, to obtain a densest subgraph of order at most k. To find the maximum weight subgraphs in $G[V \setminus H]$, we proceed analogously to the algorithm given above for Theorem 1; we omit the details. Using the size bound for k from Lemma 1(ii), we obtain the following running time.

Theorem 2. *μ-Clique can be solved in time* $O(2^h \cdot (4e \cdot (h - 1))^{h/\mu+1} \cdot h \cdot n(n + m))$*, reporting a yes-instance as no-instance with probability at most* $1/e$ *where* h *is the* h*-index of the input graph.*

Degeneracy and Dual Parameter. In this section we show that DkS is fixed-parameter tractable with respect to the combined parameter degeneracy and $\ell := n - k$, where n is the number of vertices in the input graph. Remember that in μ-Clique we fix some constant minimum density μ of the sought graph. This is necessary to bound the maximum value of k and, ultimately, obtain feasible running time bounds. For the combined parameter (d, ℓ) this constraint can be dropped. The algorithm is based on the following observation.

Lemma 3. *Let $G = (V, E)$ be a graph and let $S \subseteq V$ such that $G[S]$ is densest possible and S has size k. Then, there is no vertex in $V \setminus S$ that has degree at least $\ell + d$, where $\ell = n - k$.*

Proof. Assume that there is a vertex v of degree at least $\ell + d$ in $V \setminus S$. Since v has at most $\ell - 1$ neighbors in $V \setminus S$, it has at least $d + 1$ neighbors in S. However, because G is d-degenerate, there is a vertex u of degree at most d in $G[S]$. Thus, $G[(S \setminus \{u\}) \cup \{v\}]$ is a graph with at least one edge more than $G[S]$. This contradicts the fact that $G[S]$ is densest possible. □

Note that we can regard DkS as the problem of deleting a set of ℓ vertices whilst removing the least possible number of edges. Let us call such a vertex set *sparsest ℓ-deletion set.* To exploit Lemma 3, first mark every vertex with degree at least $\ell + d$ undeletable. Then, find a sparsest ℓ-deletion set in the degree-bounded graph induced by the deletable vertices. To find this deletion set, we employ, much in the spirit of the algorithms for maximum degree and h-index, color coding and use dynamic programming to first find connected sparsest $(\leq \ell)$-deletion sets and then combine them to an optimum one; we omit the details. By the same argument as for the algorithm for μ-Clique and maximum degree, it suffices to repeat e^{ℓ} times the random coloring and dynamic programming procedure to obtain an error probability of at most $1/e$. In summary, we have the following.

Theorem 3. Densest-k-Subgraph *can be solved in* $\mathrm{O}((4e \cdot (\ell + d - 1))^{\ell} n(n+m))$ *time, reporting a yes-instance as no-instance with probability at most $1/e$, where $\ell = n - k$ and d is the degeneracy of the input graph.*

4 Hardness Results

In this section, we present two reductions that show the limits of the approach presented above.

4.1 W[1]-hardness for Parameterization by Dual

First, we show that considering only the dual parameter ℓ leads to W[1]-hardness also in the case of μ-Clique.

Theorem 4. *For any fixed μ, $0 < \mu < 1$,* μ-Clique *is W[1]-hard with respect to the parameter $\ell = n - k$.*

Somehow counter-intuitively, the reduction used in Theorem 4 suggests that in order to obtain a graph with density μ it might be of advantage to delete a clique from the input graph. Hence, one cannot expect that the set of removed vertices induces a sparse graph. From the above reduction, we also obtain a lower bound on the running time of algorithms for μ-Clique. This bound is based on the exponential-time hypothesis (ETH) which implies that 3SAT cannot be solved in $O^*(2^{o(n)})$ time [14, 18].

Theorem 5. *μ-Clique cannot be solved in time $O^*(2^{o(\Delta/\mu)})$ for every $0 < \mu \leq 1$ unless the exponential time hypothesis (ETH) fails. Here, h is the h-index of the input instance.*

Clearly, Theorem 5 also excludes algorithms with running time $O^*(2^{o(h/\mu)})$.

4.2 W[1]-hardness for Parameterization by Degeneracy and Solution Size

Next, we show that the parameter h-index cannot be replaced by the smaller parameter degeneracy.

Theorem 6. *For any fixed μ, $0 < \mu < 1$, μ-Clique is W[1]-hard parameterized by (d, k), where d denotes the degeneracy of the input graph.*

We can use the reduction behind Theorem 6 to also exclude polynomial-size problem kernels for the parameters maximum degree and h-index.

Theorem 7. *μ-Clique does not admit polynomial-size problem kernels with respect to either maximum degree or h-index unless NP $\subseteq$ coNP/poly.*

Proof. It suffices to prove the statement for the larger maximum degree parameter. For this, we observe that the reduction used in Theorem 6 implies a cross-composition [4] from Clique into μ-Clique parameterized by maximum degree. A cross-composition from a language $L \subseteq \Sigma^*$ into a parameterized problem P is an algorithm that, given t strings $x_1, x_2, \ldots, x_t \in \Sigma^*$, computes an instance x^* of P with parameter value k such that its running time is bounded by a polynomial in $\sum_{i=1}^{t} |x_i|$, k is bounded by a polynomial in $\max_{i=1}^{t} |x_i|$, and $x^* \in P$ if and only if $x_i \in L$ for some $1 \leq i \leq t$.[2] If a parameterized problem that has a cross-composition from an NP-hard language also admits a polynomial-size problem kernel, then NP $\subseteq$ coNP/poly [4].

Let a number of instances of Clique be given and without loss of generality, assume that each instance asks for a clique of order k'.[3] Merge

[2] For readability, we simplified the more general definition of cross-composition here.

[3] If an instance asks for a smaller clique, simply add a new vertex and connect it to all other vertices of this instance.

the instances into one instance of CLIQUE by taking the disjoint union of the graphs. It is clear that this graph contains a clique of given order if and only if one of its connected components does. Then, apply the reduction used in Theorem 6 to the resulting graph. To obtain that this procedure is a cross-composition, it remains to show that the maximum degree in the created instance is bounded by a polynomial in the maximum size of the input instances. This follows since the reduction used for Theorem 6 does not merge any connected components and the introduced gadget graph has size polynomial in k'. Thus, there is cross-composition from CLIQUE into μ-CLIQUE parameterized by the maximum degree. $\square$

5 Outlook

Several research tasks remain. First, it would be interesting to improve the presented algorithms. We conjecture, however, that it is not possible to achieve a running time of $O^*(2^{o((\Delta/\mu)\log\Delta)})$, but have no proof for this at the moment. Furthermore, it would be interesting to obtain nontrivial Turing kernels for μ-CLIQUE and any of the considered parameters. Also, is there a better polynomial-time algorithm for μ-CLIQUE on planar graphs than the XP-algorithm for degeneracy? Finally, a further restriction that can be made in the area of community detection is to bound the size of the neighborhood of the μ-cliques. Efficient algorithms exploiting such bounds would be interesting and also practically relevant.

Bibliography

[1] N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4): 844–856, 1995.

[2] O. Amini, I. Sau, and S. Saurabh. Parameterized complexity of the smallest degree-constrained subgraph problem. In *Proc. 3rd IWPEC*, volume 5018 of *LNCS*, pages 13–29. Springer, 2008.

[3] B. Balasundaram, S. Butenko, and I. V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Oper. Res.*, 59(1):133–142, 2011.

[4] H. L. Bodlaender, B. M. P. Jansen, and S. Kratsch. Cross-composition: A new technique for kernelization lower bounds. In *Proc. 28th STACS*, volume 9 of *LIPIcs*, pages 165–176. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011.

[5] L. Cai. Parameterized complexity of cardinality constrained optimization problems. *Comput. J.*, 51(1):102–121, 2008.

[6] L. Cai, S. M. Chan, and S. O. Chan. Random separation: A new method for solving fixed-cardinality optimization problems. In *Proc. 2nd IWPEC*, volume 4169 of *LNCS*, pages 239–250. Springer, 2006.

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

[8] D. Eppstein and E. S. Spiro. The h-index of a graph and its application to dynamic subgraph statistics. In *Proc. 11th WADS*, volume 5664 of *LNCS*, pages 278–289. Springer, 2009.

[9] D. Eppstein, M. Löffler, and D. Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In *Proc. 21st ISAAC*, volume 6506 of *LNCS*, pages 403–414. Springer, 2010.

[10] U. Feige and M. Seltser. On the densest k-subgraph problem. Technical report, The Weizmann Institute, Department of Applied Math and Computer Science, 1997.

[11] U. Feige, D. Peleg, and G. Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[12] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.*, 18(1): 30–55, 1989.

[13] K. Holzapfel, S. Kosub, M. G. Maaß, and H. Täubig. The complexity of detecting fixed-density clusters. *Discrete Appl. Math.*, 154(11): 1547–1562, 2006.

[14] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[15] S. Khuller and B. Saha. On finding dense subgraphs. In *Proc. 36th ICALP*, volume 5555 of *LNCS*, pages 597–608. Springer, 2009.

[16] C. Komusiewicz. *Parameterized Algorithmics for Network Analysis: Clustering & Querying.* PhD thesis, Technische Universität Berlin, Berlin, Germany, 2011.

[17] S. Kosub. Local density. In *Network Analysis*, volume 3418 of *LNCS*, pages 112–142. Springer, 2004.

[18] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

[19] B. Saha, A. Hoch, S. Khuller, L. Raschid, and X.-N. Zhang. Dense subgraphs with restrictions and applications to gene annotation graphs. In *Proc. 14th RECOMB*, volume 6044 of *LCNS*, pages 456–472. Springer, 2010.