The Complexity of Detecting Fixed-Density Clusters*

Klaus Holzapfel *Sven Kosub* *Moritz G. Maaß* *Hanjo Täubig*

Institut für Informatik, Technische Universität München
Boltzmannstraße 3, D-85748 Garching b. München, Germany
{holzapfe | kosub | maass | taeubig}@in.tum.de

Abstract

We study the complexity of finding a subgraph of a certain size and a certain density, where density is measured by the average degree. Let $\gamma : \mathbb{N} \to \mathbb{Q}_+$ be any density function, i.e., γ is computable in polynomial time and satisfies $\gamma(k) \leq k-1$ for all $k \in \mathbb{N}$. Then γ-Cluster is the problem of deciding, given an undirected graph G and a natural number k, whether there is a subgraph of G on k vertices which has average degree at least $\gamma(k)$. For $\gamma(k) = k-1$, this problem is the same as the well-known clique problem, and thus **NP**-complete. In contrast to this, the problem is known to be solvable in polynomial time for $\gamma(k) = 2$. We ask for the possible functions γ such that γ-Cluster remains **NP**-complete or becomes solvable in polynomial time. We show a rather sharp boundary: γ-Cluster is **NP**-complete if $\gamma = 2 + \Omega(\frac{1}{k^{1-\varepsilon}})$ for some $\varepsilon > 0$ and has a polynomial-time algorithm for $\gamma = 2 + O(\frac{1}{k})$.

Keywords. Density-based clustering, computational complexity, graph algorithms, fixed-parameter problems.

1 Introduction

Density-based approaches are highly natural to network-clustering issues. Web communities, for instance, both well-established and emerging have in common that they show a significantly high ratio of linkage among their members (see, e.g., [17, 16, 18]). And, in VLSI layout design, collapsing subgraphs of density beyond a certain threshold into one node provides the basis for hierarchical graph-representation of large circuits to be decomposed (see, e.g., [7, 15, 1]).

The fundamental task in density-based clustering is finding a dense subgraph (cluster) of a certain size. Density of a graph might be defined in several different ways. One can define the density of an undirected graph on n vertices to be the ratio of the number of edges in the graph and the maximum edge-number of an n-vertex graph. Thus, an n-vertex-clique has density one and n isolated vertices have density zero. The definition is very intuitive and, in particular, it enables to compare graphs of different sizes directly, regarding their densities. On the other hand, since a number of edges quadratic in the number of vertices is required for a graph to be dense, small graphs are biased. Therefore, density of undirected graphs is usually measured by the average degree. A clique of size n clearly has average degree $n-1$.

*Research supported by DFG (Deutsche Forschungsgemeinschaft), grant Ma 870/6-1 (SPP 1126 Algorithmik großer und komplexer Netzwerke); Research of the third and of the fourth author supported by DFG, grant Ma 870/5-1 (Leibnizpreis Ernst W. Mayr).

The problem of deciding, given a graph G and natural numbers k and γ, if there exists a k-vertex subgraph of G having average degree at least γ, is easily seen to be **NP**-complete. In contrast to this *variable* cluster-detection problem, we focus in this paper on the *fixed-parameter* cluster-detection problem, which we call γ-Cluster. That is, we fix density-parameter γ (which generally may depend on an argument) and want to decide, given graph G and natural number k, if there exists a k-vertex subgraph of G with average degree at least γ (or $\gamma(k)$, more generally). We are interested in what choices of γ still admit polynomial-time algorithms, and for which γ the problem becomes **NP**-complete.

Studying the complexity of the fixed-parameter problem is motivated by at least two reasons. First, knowing the precise boundary between polynomial-time and **NP**-complete cases is essential to obtain efficient methods for the above-mentioned clustering issues in several settings, e.g., Web graphs, where good choices of γ describe reality sufficiently. Second, if the polynomial-time cases can be realized by a uniform algorithm (i.e., parameters $t \leq \gamma$ may be given to the input), then we can approximate the maximum average degree reachable on n vertices in a graph within factor $\frac{n}{\gamma}$. The best algorithm known guarantees approximation within a factor $n^{\frac{1}{3}-\varepsilon}$ for some $\varepsilon > 0$ [10]. Thus, we would outperform this algorithm if we find a (uniform) polynomial-time algorithm for γ-Cluster up to little over $n^{\frac{2}{3}+\varepsilon}$. Unfortunately, the boundary with the **NP**-complete cases turns out to be much lower.

Previous Work

The problem of finding dense subgraphs has attracted a lot of attention in the context of combinatorial optimization.

Gallo, Grigoriadis, and Tarjan [12] showed, by using flow techniques, that there is a polynomial-time algorithm for the densest subgraph problem, in which we are supposed to find a subgraph of arbitrary size with highest average degree. Feige, Kortsarz, and Peleg [10] studied a restricted version, which they called the dense k-subgraph problem, where we have to find a subgraph with highest average degree among all subgraphs on k vertices. They provide a polynomial-time algorithm that approximates the maximum average degree of such k-vertex subgraphs within factor $n^{\frac{1}{3}-\varepsilon}$ for some $\varepsilon > 0$. Several authors proved approximation results for the dense k-subgraph problem using different techniques, mainly greedy algorithms [4, 6] and semidefinite programming [11, 20]. For special graph classes, they obtained partly better approximation results. Arora, Karger, and Karpinski [2] showed that the dense k-subgraph problem on dense graphs (i.e., with quadratic number of edges) admits a polynomial-time approximation scheme, which has been improved by Czygrinow to a fully polynomial-time approximation scheme [8]. In contrast to this, it is not known whether the dense k-subgraph problem on general graphs is not approximable within factor $(1+\varepsilon)$ for all $\varepsilon > 0$ (unless $\mathbf{P} = \mathbf{NP}$ or similar complexity-theoretic collapses), although it is even conjectured that the problem is hard to approximate within factor n^{ε} for some $\varepsilon > 0$ [10].

Fixed-parameter problems were also considered in our setting. Nehme and Yu [19] investigated the complexity of the constrained maximum value sub-hypergraph problem, which contains the dense k-subgraph problem as a special case. They obtained bounds on the number of (hyper-)edges a (hyper-)graph may have, such that the problem is still polynomial-time solvable, namely, $n - s + \alpha \log n$, where n is the number of vertices, s the number of connected components, and α any constant. Similar fixed parameter-restrictions to the input graphs were also considered in [4, 3]. Note that this scenario has no consequences for our problem since these restrictions affect the graph outside of possible dense subgraphs, and we are interested in the existence of dense subgraphs of fixed quality inside an arbitrary graph.

Most recently, Asahiro, Hassin, and Iwama [3] studied the k-$f(k)$ dense subgraph problem, $(k, f(k))$-DSP for short, which asks whether there is a k-vertex subgraph of a given graph G which has at least $f(k)$ edges. This problem is almost the same problem as our γ-cluster problem, since obviously, a k-$f(k)$ subgraph has average degree at least $\frac{2f(k)}{k}$. The authors proved that the problem remains **NP**-complete for $f(k) = \Theta(k^{1+\varepsilon})$ for all $0 < \varepsilon < 1$ and is polynomial-time solvable for $f(k) = k$. From this results we can conclude that γ-Cluster is **NP**-complete for $\gamma = \Theta(k^{\varepsilon})$ for any $0 < \varepsilon < 1$, and is decidable in polynomial time for $\gamma = 2$. Feige and Seltser [11] even proved that $(k, f(k))$-DSP is **NP**-complete if $f(k) = k + k^{\varepsilon}$ (which, in our notation, is $\gamma = 2 + 2k^{\varepsilon - 1}$) for any $0 < \varepsilon < 2$. We will enhance those bounds to more general settings.

This Work

In this paper we show that γ-Cluster is polynomial-time solvable for $\gamma = 2 + O(\frac{1}{k})$ and that γ-Cluster is **NP**-complete for $\gamma = 2 + \Omega(\frac{1}{k^{1-\varepsilon}})$ for $0 < \varepsilon < 2$. We thus establish a rather sharp boundary between polynomial time solvable and **NP**-complete cases. As a corollary we obtain, for the more intuitive case of γ constant, that detecting a k-vertex subgraph of average degree at least two (which is nearly the case of any connected graph) can be done in polynomial time whereas finding a k-vertex subgraph of slightly-higher average degree at least $2 + \varepsilon$ is already **NP**-complete. Thus, density-based clustering is inherently hard as a general methodology.

In terms of the $(k, f(k))$-DSP our results mean that $(k, f(k))$-DSP remains **NP**-complete if $f(k) = k + \Omega(k^{\varepsilon})$ for any $0 < \varepsilon < 2$, which, for $\varepsilon \leq 1$, is more precise than in [3, 11], and is polynomial-time decidable for $f(k) = k + c$ for all (constant) integers c.

The proof of the polynomial-time cases is mainly based on dynamic programming over collections of minimal subgraphs having certain properties. For instance, for the above-mentioned polynomial-time result for $(k, f(k))$-DSP with $f(k) = k$ [3], we simply need to find shortest cycles in a graph, which is easy. For functions $f(k) = k + c$ with $c > 0$, the search for similar minimal subgraphs is not obvious to solve and is the main difficulty to overcome in order to obtain polynomial-time algorithms. In the **NP**-hardness proofs we adapt techniques used by [3, 11], that are well suited for Θ-behavior of functions but lead to different reductions according to the different growth classes. Thus the main issue for getting results for Ω-behavior is to unify reductions by a non-trivial choice of the parameters involved.[1]

2 Definitions and Main Results

Throughout this paper we consider undirected graphs without loops. Let G be any graph. $V(G)$ denotes the set of vertices of G and $E(G)$ denotes the set of edges of G. The size of a graph is $|V(G)|$, i.e., the cardinality of $V(G)$. For any function $\gamma : \mathbb{N} \to \mathbb{Q}_+$, graph G is said to be

[1] Basically, having Turán's theorem [21] in mind, one could ask whether it is possible, at least in the case of dense graphs, to deduce intractability results using inapproximability of Maximum Clique due to Håstad [14]: there is no polynomial-time algorithm finding cliques of size at least $n^{\frac{1}{2}+\varepsilon}$ (where n is the size of the maximum clique) unless $\mathbf{P} = \mathbf{NP}$. Assume we would have a polynomial-time algorithm for γ-Cluster with, e.g., $\gamma(k) = \beta\binom{k}{2}$ and $0 < \beta < 1$, are we now able to decide whether there is a clique of size $k^{\frac{1}{2}+\varepsilon}$? Turán's theorem [21] says that there is a clique of size k in a graph with n vertices and m edges, if $m > \frac{1}{2}n^2\frac{k-1}{k-2}$. Unfortunately this implies that we can only assure that in a graph with n vertices and at least $\beta\binom{n}{2}$ edges, there is a clique of size at most $\frac{3-2\beta}{1-\beta}$, which is constant and makes the argument fail.

a γ-*cluster* if and only if $d(G) \geq \gamma\big(|V(G)|\big)$ where $d(G)$ denotes the average degree of G, i.e., $d(G) = 2|E(G)|/|V(G)|$.

We study the complexity of the following problem. Let $\gamma : \mathbb{N} \to \mathbb{Q}_+$ be any function.

Problem:	γ-CLUSTER
Input:	A graph G and a natural number k
Question:	Does G contain a γ-cluster of size k?

Note that 0-CLUSTER is a trivial problem and that $(k-1)$-CLUSTER = CLIQUE. Moreover, it is easily seen that γ-CLUSTER is in **NP** whenever γ is computable in polynomial time. The following theorem expresses our main results on detecting fixed-density clusters.

Theorem 1. *Let $\gamma : \mathbb{N} \to \mathbb{Q}_+$ be computable in polynomial time, $\gamma(k) \leq k-1$.*

1. *If $\gamma = 2 + O(\frac{1}{k})$, then γ-CLUSTER is solvable in polynomial time.*
2. *If $\gamma = 2 + \Omega(\frac{1}{k^{1-\varepsilon}})$ for some $\varepsilon > 0$, then γ-CLUSTER is **NP**-complete.*

In the remainder of the paper we prove Theorem 1. Section 3 contains the polynomial-time cases. Section 4 establishes the **NP**-completeness statements of Theorem 1.

3 Computing $(2 + O(\frac{1}{k}))$-dense Subgraphs in Polynomial Time

In this section we show how to solve γ-CLUSTER for $\gamma = 2 + O(\frac{1}{k})$ in polynomial time. In other words, we prove that searching a k-vertex subgraph with at least $k + c$ edges with c constant is a polynomial-time problem. We will formalize this issue in the problem EXCESS-c SUBGRAPH.

For a graph G, let the *excess of* G, denoted by $\nu(G)$, be defined as $\nu(G) = |E(G)| - |V(G)|$. A (sub)graph G with $\nu(G) \geq c$ is said to be an *excess-c (sub)graph.*

Problem:	EXCESS-c SUBGRAPH
Input:	A graph G and natural number k
Question:	Does G contain an excess-c subgraph of size k?

We will show how to find excess-c subgraphs in polynomial time. The general solution is based on the case of a connected graph which is handled by the following lemmas:

Lemma 2. *Let $c \geq 0$ be any integer. Given a connected graph G on n vertices, an excess-c subgraph of minimum size can be computed in time $O(n^{2c+2})$.*

Proof. Let G be any connected graph with $\nu(G) \geq c$. Then there exists a subgraph G_c of minimum size with excess c. For the degree-sum of G_c we obtain

$$\sum_{v \in V(G_c)} \deg_{G_c}(v) = 2|E(G_c)| = 2(|V(G_c)| + c).$$

Since G_c is minimal subject to the number of vertices, there exists no vertex with degree less than two. Therefore the number of vertices with degree greater than two in G_c is at most

$$\sum_{v \in V(G_c)} (\deg_{G_c}(v) - 2) = 2(|V(G_c)| + c) - 2|V(G_c)| = 2c.$$

Let S be the set of all vertices with degree greater than two in G_c. If there is a path connecting vertices $u, v \in S$ using only vertices from $V(G_c) \setminus S$ (u and v are not necessarily distinct), then there can be no shorter path connecting u and v containing vertices from $V(G) \setminus V(G_c)$. Otherwise G_c would not be minimal subject to the number of vertices. In the following we will describe how to find such a subgraph G_c if it exists.

We examine all sets $S' \subseteq V(G)$ of size at most $2c$ such that S' contains only vertices with degree greater than two in G, i.e., the elements of S' are those vertices where paths can cross. For such a set we can iteratively construct a candidate $H(S')$ for G_c. In each step we include a path which has minimum length among all paths that connect any two vertices in S'. We may restrict ourselves to those paths that do not intersect or join common edges, since otherwise $H(S')$ can be also obtained by one of the other possible choices of a set S'. This process is done until either excess c is reached or no further connecting path exists. In the latter case the set S' does not constitute a valid candidate for G_c. Otherwise $H(S')$ is kept as a possible choice for G_c. After considering all possibilities for S', the graph G_c can be chosen as a vertex-minimal subgraph among all remaining candidates. Note that G_c is not unique with respect to exchanging paths of the same length.

Since, $|S'| \leq 2c$, there are

$$\sum_{i=1}^{2c} \binom{n}{i} = O(n^{2c})$$

possible choices for S'. For the verification of a chosen set S' consisting of i vertices we have to find iteratively $i + c$ shortest non-crossing paths, e.g., by using $i + c \leq 3c$ parallel breadth-first-search runs, which takes time $O(3c|E(G)|) = O(n^2)$.

Finally, this implies that determining an excess-c subgraph of minimum size by testing all possible choices of S' can be done in total time $O(n^{2c+2})$. Note that for $c = 0$ we only have to find a shortest cycle (e.g., by breadth-first search) which can be done in time $O(n^2)$. ❑

Unfortunately the algorithm of Lemma 2 cannot directly be used for the general case of possibly non-connected graphs. For those graphs vertices from different connected components may be chosen. Therefore our algorithm is based on solving the subproblem of maximizing the excess for a given number of vertices within a connected graph.

Lemma 3. *Let $c \geq 0$ be any integer. Given a connected graph G with n vertices. Let ν_i be the maximum excess of an i-vertex subgraph of G. Calculating $\min\{\nu_i, c\}$ for all values of $i \in \{0, 1, \ldots, n\}$ can be done in time $O(n^{2c+2})$.*

Proof. There are some basic observations. First of all, $\nu_0 = 0$. Also, since G is connected, $\nu_i \geq -1$ for all $i \in \{1, 2, \ldots, n\}$. Furthermore, due to the connectivity of G the subgraph can iteratively be extended without decreasing the excess. Thus, if there exists a subgraph on i vertices having excess ν_i, the value ν_i is a lower bound for the maximum excess of subgraphs with more vertices. Therefore it is sufficient to know the minimum number of vertices necessary to achieve excess c (as done in Lemma 2).

The maximum excess we are interested in is bounded from above by c. We get the minimum number of vertices needed for all possible values of $\nu \in \{0, 1, \ldots, n\}$ by performing $c+1$ iterations of the algorithm of Lemma 2. Using these results we can easily calculate for each $i \in \{0, 1, \ldots, n\}$ the desired value $\min\{\nu_i, c\}$. This takes total time $O(n^{2c+2})$. ❑

Before we proceed to the main theorem we have to discuss a further property. Let (G, k) be the instance of the Excess-c Subgraph problem, i.e., we have to find a subgraph of G on k vertices with at least $k + c$ edges. In linear time we can (as a preprocessing step) partition G into its connected components and calculate their excess. Let $C_1, \ldots, C_r$ be the list of the components, sorted non-increasingly by their excess. Note that $\nu(C_j) \geq -1$ since all components are connected. Let j_0 denote the maximum index of the components with non-negative excess and k_0 the total number of all vertices of those components.

Lemma 4. *1. If $k > k_0$ then there is a maximum excess subgraph comprising all vertices from the non-negative excess components $C_1, \ldots, C_{j_0}$.*

2. If $k \leq k_0$ then there always exists a subgraph of size k having maximum excess within G and consisting only of vertices from components with non-negative excess.

Proof. Let G' be an induced subgraph of G. Assume that G' contains vertices of a component C_i with negative excess while there exists a component C_j with positive excess that is not contained entirely.

If at least one vertex in C_j is selected, there exists another so far not selected vertex v in C_j that is adjacent to G'. Since C_i must be a tree, there must exist a selected vertex $u \in C_i$ that is a leaf in the selection, i.e., it is incident to at most one edge in G'. By exchanging u and v, no excess is lost.

Otherwise, no vertex in C_j is selected. Once again we exchange leaves from C_i with connected vertices from C_j. There are two possibilities. First, if C_j is selected entirely, we cannot lose excess because $\nu(C_j) \geq 0$. Second, if all vertices of C_i were exchanged, once again we cannot lose excess since $\nu(C_i) = -1$ and C_j is connected (and thus has excess at least -1).

This process can be iterated until there are no vertices selected from components with negative excess or all components with positive excess are contained entirely. ❑

With these results we are able to state the main theorem of this section.

Theorem 5. *Let c be any integer.* Excess-c Subgraph *can be decided in time $O(n^{2|c|+4})$.*

Proof. Let c be any fixed integer. Let (G, k) be a problem instance. The problem can be divided into two cases.

In the first case, where $k \geq k_0$, the problem can be solved straightforward. Because of Lemma 4, there exists a maximum-excess subgraph on k vertices that contains all components with non-negative excess entirely. Therefore all remaining vertices must be chosen from the components with negative excess. Those components are trees (each having excess -1 by definition) and thus the selected vertices within these components induce a forest. Since we want to maximize the excess, we have to minimize the number of trees. Therefore, as long as possible, we choose complete components ordered by non-increasing size (i.e., largest trees first). From the next component we choose a subtree of sufficient size, to get exactly the desired number of vertices. This procedure determines the minimum number of trees to choose. Finally, the maximum excess of a subgraph of G on k vertices can be evaluated by adding up the excess of all used components. Obviously, in this case the time bound of the theorem holds.

In the other case, $k < k_0$, we may restrict our choice to those components with non-negative excess. We show that it is sufficient to calculate separately for each such component the minimum size of subgraphs for all values of excess within the fixed range $\{0, 1, \ldots, c + 1\}$. The original

```
initialize the arrays X and Y
for all j ∈ {1, ..., j_0} do
   for all i ∈ {1, ..., min(|V(C_j)|, k)} do
      for all l ∈ {0, ..., k − i} do
         if Y[l + i] < X[l] + A_j[i] then
            Y[l + i] = X[l] + A_j[i]
   copy array Y to X
```

Figure 1: Algorithm for excess-aggregation of connected components.

problem can be decided by combining these solutions. For each component C_j we create an array A_j. At index $i \in \{0, 1, \ldots, |V(C_j)|\}$ we will store the maximum excess for any (induced) subgraph of component C_j on i vertices. As we will see later values larger than $c+1$ are of no interest. In these cases the lower bound $c+1$ will be used instead. Note that the maximum excess cannot decrease with larger induced subcomponents, because we can simply add vertices that are connected to the subgraph. Due to Lemma 3 array A_j can be calculated in time $D|V(C_j)|^{2(|c|+1)+2}$ for some $D > 0$. Hence, the total time to calculate the values for A_j for all components is

$$\sum_{j=1}^{r} D|V(C_j)|^{2(|c|+1)+2} \leq D\left(\sum_{j=1}^{r} |V(C_j)|\right)^{2|c|+4} = D|V|^{2|c|+4}.$$

Based on the results of the calculation we can distinguish two different cases.

- If there exists a component that contains an excess-$(c+1)$ subgraph on $k_1 \leq k$ vertices, we can choose this subgraph and add a sufficient number $k - k_1$ of vertices such that the excess decreases by at most one. This can be achieved by appending remaining vertices of the component, adding entire so far unused components (with excess $\nu \geq 0$) and adding at most one incomplete component (a connected subgraph with excess $\nu \geq -1$).

- Otherwise, for the second case, all excess-$(c+1)$ subgraphs of any component have more than k vertices. Assuming that we already calculated the values of the arrays A_j, we can compute the maximum excess of a k-vertex subgraph from G by considering suitable subsets of the components. Therefore we have to decide how many vertices of each component have to be selected.[2]

  From each component C_j at most $\min(|V(C_j)|, k)$ vertices can be selected. Remember that the corresponding maximum excess is stored in A_j. We iterate over all components and within the components over all possible subgraph-sizes and store the currently best sub-result in array X. Thus after each iteration $X[i]$ contains the maximum possible excess for an i-vertex subgraph of the so far processed components. Finally $X[k]$ contains the value of the maximum excess for any subgraph on k vertices. Thus, it can be decided whether there exists an excess-c subgraph of size k. Figure 1 shows an algorithm for this calculation in pseudo-code.

[2]Note that this problem is a variant of Subset Sum, using a set of integer-intervals $\big\{\{0, 1, \ldots, |V(C_j)|\} \;\big|\; 0 \leq j \leq j_0\big\}$. Despite Subset Sum is **NP**-complete, this problem can be solved in polynomial time, because of the present unary representation of n.

Since the total size of all components is bounded by n (first and second loop) and $k \leq n$ (third loop) the total calculation cost is $O(n^2)$. ❑

So far we only considered the EXCESS-c SUBGRAPH problem for constant values c. If we are interested in a k-vertex subgraph with excess $f(k) = O(1)$, the same method can be applied. From $f = O(1)$ we know that $f(k)$ is bounded from above by a constant c'. Obviously the time complexity for our algorithm is $O(n^{2c'+4})$, if $f(k)$ can be computed in the same time. This problem corresponds to finding a $(k + O(\frac{1}{k}))$-dense subgraph. Applying some modifications the method can be used to find such a subgraph instead of only deciding its existence.

Corollary 6. *For polynomial-time computable $\gamma = 2 + O(\frac{1}{k})$, γ-CLUSTER is is solvable in polynomial time and, moreover, finding a γ-cluster is solvable in polynomial time.*

Similarly, the problem can be examined for $f = O(\log k)$ which leads to a quasi-polynomial time algorithm.

Corollary 7.
1. *For polynomial-time computable $\gamma = 2 + O(\frac{\log k}{k})$, finding γ-clusters can be done in time $n^{O(\log n)}$.*
2. *Let $\gamma = 2 + \Theta(\frac{\log k}{k})$ be polynomial-time computable. If γ-CLUSTER is **NP**-complete, then **NP** $\subseteq$ **DTIME**$(n^{O(\log n)})$.*

4 Finding $(2 + \Omega(\frac{1}{k^{1-\varepsilon}}))$-dense Subgraphs is NP-complete

In this section we prove that all γ-CLUSTER problems are complete for **NP** if $\gamma = 2 + \Omega(\frac{1}{k^{1-\varepsilon}})$ for some $\varepsilon > 0$. In doing so, we focus on the $(k, f(k))$-DSP, namely, we show that $(k, f(k))$-DSP is **NP**-complete whenever $f = k + \Omega(k^\varepsilon)$.

For this, we need the concept of a quasi-regular graph. A graph G is said to be *quasi-regular* if and only if the difference between the maximal and the minimal degree of the vertices in G is at most one.

Proposition 8. *For every $n \geq 0$ and $0 \leq m \leq \binom{n}{2}$ both given in unary (i.e., as inputs 1^n and 1^m), a quasi-regular graph having exactly n vertices and m edges can be computed in time polynomial in the input length.*

Proof. Define $d^* =_{\text{def}} \lceil \frac{2m}{n} \rceil$ and $d_* =_{\text{def}} \lfloor \frac{2m}{n} \rfloor$. Then there are two distinct cases: Either d^* is even or d_* is even. First, let d_* be even. Compute a d_*-regular graph (by considering n vertices to be circularly ordered and connecting each vertex with its $d_*/2$ left and its $d_*/2$ right neighbors in the circuit) and add a matching of size $m - (d_*/2)n$. This is possible since

$$m - \frac{d_* n}{2} \leq m - \left(\frac{2m}{n} - 1\right)\frac{n}{2} = \frac{n}{2}.$$

If d^* is even then compute a d^*-regular graph and remove an existing matching of size $(d^*/2)n - m$. Analogously to the other case this is possible since

$$\frac{d^* n}{2} - m \leq \left(\frac{2m}{n} + 1\right)\frac{n}{2} - m = \frac{n}{2}.$$

Clearly the graphs can be computed in time polynomial in the values of the inputs, and hence polynomial in the length of the unary representations of n and m. ❑

Theorem 9. *Let $f : \mathbb{N} \to \mathbb{N}$ be a polynomial-time computable function such that $f = k + \Omega(k^\varepsilon)$ for some $\varepsilon > 0$ and $f(k) \le \binom{k}{2}$. Then, $(k, f(k))$-DSP is* **NP***-complete.*

Proof. Let f be a polynomial-time computable function with $f = k + \Omega(k^\varepsilon)$ for some rational $\varepsilon > 0$ and $f(k) \le \binom{k}{2}$. Containment of $(k, f(k))$-DSP in **NP** is obvious. We prove the **NP**-hardness of $(k, f(k))$-DSP by reduction from a special version of CLIQUE which will be explained below. Since there are several cases to be handled we need different constructions. However, in each of this constructions the following three operations (with parameters from $\mathbb{N}$) on graphs are involved (exactly ordered as listed below).

- R_s: Let G be any undirected graph. Define the following sequence of graphs: $G_0 =_{\text{def}} G$ and, for $j > 0$, $G_j =_{\text{def}} h(G_{j-1})$ where h transforms a graph I by adding to I a new vertex which has an edge to each vertex in I. Define $R_s(G) =_{\text{def}} G_s$. Obviously, the following property holds:

$$G \text{ has a clique of size } k \iff R_s(G) \text{ has a clique of size } k + s. \tag{1}$$

  The operator R_s can be used to define a special **NP**-complete version of CLIQUE (see, e.g., [3]). Define $\text{CLIQUE}_{\frac{1}{2}}$ to be the set of all instances (G, k) such that G has a clique of size k and it holds $|V(G)| \le (1 + \frac{1}{2})k$. It is easily seen that CLIQUE can be reduced to $\text{CLIQUE}_{\frac{1}{2}}$, namely by applying R_s to a graph G with parameter $s = 2|V(G)| - 3k$ for each instance (G, k) with $|V(G)| > (1 + \frac{1}{2})k$. The transformed graph G_s now has $|V(G)| + s = \frac{1}{2}s + (1 + \frac{1}{2})k + s = (1 + \frac{1}{2})(k + s)$ vertices, and using (1) the new clique-size G_s is asked for, is $k + s$.

- S_t: Let G be any undirected graph. S_t transforms G to a graph G_t by replacing each edge in G by a path of length $t + 1$ involving t new vertices. The new vertices are referred to as *inner vertices* and the old vertices are referred to as *outer vertices*. Note that inner vertices always have degree 2 and that an outer vertex has equal degrees in G_t and in G. It is easily seen that cliques in G of size $k \ge 3$ are related to subgraphs of G_t as follows (for formal proofs, see, e.g., [9, 13, 11]):

$$G \text{ has a clique of size } k \iff S_t(G) \text{ has a subgraph with } k + t\binom{k}{2} \text{ vertices and } (t+1)\binom{k}{2} \text{ edges.} \tag{2}$$

- $T^\alpha_{r,N(r)}$ (with $\alpha \in \{0, 1\}$): Let G be any undirected graph. $T^0_{r,N(r)}$ transforms G by the disjoint union with a quasi-regular graph $A(r, N(r))$ with r vertices and $N(r)$ edges. In transformations by $T^1_{r,N(r)}$, we additionally have edges between each vertex in $A(r, N(r))$ and each vertex in G.

Note that all these operations are monotonic, i.e., if G is a subgraph of H, then the transformed graph of G is a subgraph of H transformed by the same operations with same parameters.

Let us first consider an arbitrary function $f : \mathbb{N} \to \mathbb{N}$ in order to explain the general outline of the proof. Let (G, k) be any instance to $\text{CLIQUE}_{\frac{1}{2}}$ with $|V(G)| \le \frac{3}{2}k$. We will construct a new graph G' such that G has a clique of size k if and only if G' has a subgraph of size k' with at least $f(k')$ edges. Define $G' =_{\text{def}} (T^\alpha_{r,N(r)} \circ S_t \circ R_s)(G)$[3] and let the parameters s, t, r, α be

[3]Our usage of $f \circ g$ is $(f \circ g)(x) =_{\text{def}} f(g(x))$.

fixed. The parameter $N(r)$ will be defined as

$$N(r) =_{\text{def}} f(k+s+t\binom{k+s}{2}+r) - (t+1)\binom{k+s}{2} - \alpha r\left(k+s+t\binom{k+s}{2}\right).$$

Suppose for the moment that $N(r) \geq 0$. We have to prove two cases.

For the first case, assume that G has a clique of size k. Let C be such a clique. We consider the graph C' defined as $C' =_{\text{def}} (T^{\alpha}_{r,N(r)} \circ S_t \circ R_s)(C)$. Thus, C' is a subgraph of G' and we have

$$|E(C')| = (t+1)\binom{k+s}{2} + N(r) + \alpha r\left(k+s+t\binom{k+s}{2}\right).$$

Hence, C' is a subgraph of G' of size $k+s+t\binom{k+s}{2}+r$ with at least $f(k+s+t\binom{k+s}{2}+r)$ edges.

For the second case, assume that G does not contain a clique of size k. We must show that in this case each subgraph of G with $k+s+t\binom{k+s}{2}+r$ vertices has less than $f(k+s+t\binom{k+s}{2}+r)$ edges. To do so, we first determine a subgraph with maximum number of edges among all subgraphs of G' on $k+s+t\binom{k+s}{2}+r$ vertices. In particular, we will guarantee that there exists such a subgraph containing entirely the graph $A(r, N(r))$. Let H be the induced subgraph of G' containing no vertices of $A(r, N(r))$. Let $l = k+s+t\binom{k+s}{2}$. Let X be any induced subgraph of G' having $l+r$ vertices. Thus there is a z with $0 \leq z \leq r$ (more precisely, $z \leq \min\{r, |V((S_t \circ R_s)(G))| - l\}$) such that $l+z$ vertices of X belong to H and $r-z$ vertices of X belong to $A(r, N(r))$. Let Y be an induced subgraph which results from replacing, in a certain way, z vertices in X which belong to H by the remaining z vertices of $A(r, N(r))$ which are not in X. Note that, if $t \geq 1$, then the vertices in H can be iteratively removed in such an order that always a vertex with degree at most 2 is removed (simply by removing first all the inner vertices around an outer vertex and then the outer vertex which now has degree 0). We are done if we can show that $|E(Y)| \geq |E(X)|$ since, by using Properties (1) and (2), we can argue as follows. Let J be a subgraph of H of size l with maximum number of edges. Then the graph $C' =_{\text{def}} T^{\alpha}_{r,N(r)}(J)$ has the maximum number of edges among all subgraphs of G' on k' vertices. For the number of edges of C' we easily obtain

$$\begin{aligned}
|E(C')| &= |E(J)| + N(r) + \alpha r l \quad \leq \quad (t+1)\binom{k+s}{2} - 1 + N(r) + \alpha r\left(k+s+t\binom{k+s}{2}\right) \\
&= f(k+s+t\binom{k+s}{2}+r) - 1.
\end{aligned}$$

Hence, G' does not contain a subgraph with $k+s+t\binom{k+s}{2}+r$ vertices and at least $f(k+s+t\binom{k+s}{2}+r)$ edges.

In order to make these arguments work we have to choose all the parameters such that the following two conditions can be satisfied:

- *Constructibility:* We have to guarantee that the graph G' can be computed in polynomial time. Obviously the operations R_s and S_t are polynomial computable if parameters s and t can be computed in polynomial time. We further have to show that the graph $A(r, N(r))$ exists and can be computed in polynomial time. The latter condition is assured by Proposition 8 since r will depend polynomially on k which is logarithmic in the size of the graph, thus a unary description of r can be computed in polynomial time. It remains to show that $N(r) \geq 0$ and $N(r) \leq \binom{r}{2}$. Usually, $N(r) \geq 0$ is easily seen and it is often proved together with the next condition.

- *Exchangeability:* This condition refers to the claim $|E(Y)| \geq |E(X)|$ used above. Note that the claim is trivial for $z = 0$. For $z \geq 1$ we consider the edge balance of transforming X into Y. In the case of $\alpha = 0$, which is the majority of our cases, we will argue as follows. On the one hand, we remove at most Δz edges from H, for some Δ. On the other hand we add at least $\frac{1}{2}\lfloor\frac{2N(r)}{r}\rfloor z$ edges in $A(r, N(r))$ It is thus sufficient to satisfy that $\frac{1}{2}\lfloor\frac{2N(r)}{r}\rfloor z \geq \Delta z$ or, if $2\Delta \in \mathbb{N}$, equivalently, $\frac{N(r)}{r} \geq \Delta$. In the cases with $\alpha = 1$ we will employ more refined arguments.

Since $f = k + \Omega(k^\varepsilon)$ for some rational $\varepsilon > 0$, there exists a natural number $D > 1$ such that for some $k_0 \in \mathbb{N}$, $k + D^{-1}k^\varepsilon \leq f(k)$ for all $k \geq k_0$. Obviously, we may suppose that $\varepsilon < \frac{1}{8}$ and $D \geq 5$. Since we will have to respect several finer growth-classes the function f might belong to, we choose one argument k' to distinguish between these different classes. Define

$$k' =_{\text{def}} \left\lceil \left(D^6k^2\right)^{\frac{1}{\varepsilon}} \right\rceil .$$

Clearly, k' is computable in time polynomial in the length of k. Depending on the function value $f(k')$ we choose different parameters s, t, r, and α, such that $k' = k + s + t\binom{k+s}{2} + r$ to obtain the graph G'. We distinguish between five cases that represent a partitioning of the interval between $k' + D^{-1}k'^\varepsilon$ and $\binom{k'}{2}$.

Case I. Let $k' + D^{-1}k'^\varepsilon \leq f(k') < k' + Dk'$. We split this case in several subcases. We consider, depending on j with $0 \leq j < \log_{\frac{7}{6}} \frac{1}{\varepsilon}$, the ranges $k' + D^{-1}k'^{\left(\frac{7}{6}\right)^j\varepsilon} \leq f(k') \leq k' + Dk'^{\left(\frac{7}{6}\right)^{j+1}\varepsilon}$. Clearly, we can combine those subcases to cover the complete range from $k' + D^{-1}k'^\varepsilon$ to $k + Dk'$ as required for Case I. For each value of j we apply R_s, S_t, and $T^\alpha_{r,N(r)}$ with the following parameters:

$$s = 0, \qquad t = (k' - r - k)/\binom{k}{2}, \qquad \alpha = 0,$$

$$r = \left\lceil (4D^4k^2)^{\left(\frac{7}{6}\right)^j} \right\rceil + \left[\left(k' - \left\lceil (4D^4k^2)^{\left(\frac{7}{6}\right)^j} \right\rceil - k \right) \mod \binom{k}{2} \right]$$

Note that $t \in \mathbb{N}$ by the modular term in the definition of r. Trivially, we have $k + s + t\binom{k+s}{2} + r = k + t\binom{k}{2} + r = k'$. For k large enough (and thus k' and r as well) constructibility and exchangeability can be satisfied as can be seen by the following calculations.

- *Constructibility:*

$$\begin{aligned} N(r) &\leq k' + Dk'^{\left(\frac{7}{6}\right)^{j+1}\varepsilon} - (t+1)\binom{k}{2} \quad = \quad r + Dk'^{\left(\frac{7}{6}\right)^{j+1}\varepsilon} - \binom{k}{2} + k \\ &\leq r + D\left(2D^6k^2\right)^{\frac{7}{6}\left(\frac{7}{6}\right)^j} \quad \leq \quad r + \left(3D^8k^3\right)^{\left(\frac{7}{6}\right)^j} \quad \leq \quad \left((2D^4k^2)^{\left(\frac{7}{6}\right)^j}\right)^2 \\ &\leq \left(\frac{1}{2}r\right)^2 \quad \leq \quad \binom{r}{2} \end{aligned}$$

- *Exchangeability:* Since $t \geq 1$ for $k > 0$, we can choose $\Delta = 2$ and we obtain the following:

$$\begin{aligned} N(r) &\geq k' + D^{-1}k'^{\left(\frac{7}{6}\right)^j\varepsilon} - (t+1)\binom{k}{2} \quad \geq \quad r + D^{-1}\left((D^6k^2)^{\frac{1}{\varepsilon}}\right)^{\left(\frac{7}{6}\right)^j\varepsilon} - \binom{k}{2} + k \\ &\geq r + (D^5k^2)^{\left(\frac{7}{6}\right)^j} - k^2 \quad \geq \quad 2r \end{aligned}$$

Case II. Let $k + Dk' = (1+D)k' \le f(k') < (1+D)k'^{\frac{3}{2}}$. Apply R_s, S_t, and $T^{\alpha}_{r,N(r)}$ with the following parameters ($\varepsilon < \frac{1}{8}$):

$$s = 0, \qquad t = 1, \qquad \alpha = 0, \qquad r = k' - \binom{k}{2} - k$$

Clearly, we have $k + s + t\binom{k+s}{2} + r = \binom{k+1}{2} + r = k'$. For k large enough (and thus k', r as well), constructibility and exchangeability can be satisfied as can be seen by the following calculations.

- *Constructibility:*

$$N(r) \;\le\; f(k') \;\le\; (1+D)\left(r + \binom{k+1}{2}\right)^{\frac{3}{2}} \;\le\; \frac{3}{2}D(2r)^{\frac{3}{2}} \;\le\; \frac{9}{2}Dr^{\frac{3}{2}} \;\le\; \frac{1}{4}r^2 \;\le\; \binom{r}{2}$$

- *Exchangeability:* Since $t = 1$ we can choose $\Delta = 2$ and we have the following:

$$N(r) \;\ge\; (1+D)\left(r + k + \binom{k}{2}\right) - 2\binom{k}{2} \;\ge\; (1+D)r + (D-1)\binom{k}{2} \;\ge\; 2r.$$

Case III. Let $(1+D)k'^{\frac{3}{2}} \le f(k') < \binom{k'}{2} - k'^{\frac{9}{8}}$. Apply R_s, S_t, and $T^{\alpha}_{r,N(r)}$ with the following parameters:

$$s = 0, \qquad t = 0, \qquad \alpha = 0, \qquad r = k' - k$$

Obviously, $k + s + t\binom{k+s}{2} + r = k + r = k'$. Note that since $\varepsilon < \frac{1}{8}$ we have $k^2 \le k'^{\frac{1}{8}}$. For k large enough (and thus k', r as well), constructibility and exchangeability can be satisfied as can be seen by the following calculations.

- *Constructibility:*

$$N(r) \;\le\; \binom{k+r}{2} - k'^{\frac{9}{8}} - \binom{k}{2} \;\le\; \binom{r}{2} + k'(k - k'^{\frac{1}{8}}) \;\le\; \binom{r}{2}$$

- *Exchangeability:* Since G has at most $\frac{3k}{2}$ vertices, we can set $\Delta = \frac{3k}{2} - 1$ (observe that $2\Delta \in \mathbb{N}$) and we obtain the following:

$$N(r) \;\ge\; (1+D)(k+r)^{\frac{3}{2}} - \binom{k}{2} \;\ge\; r\left((1+D)(k+r)^{\frac{1}{2}} - 1\right) \;\ge\; r(k^{\frac{1}{\varepsilon}} - 1) \;\ge\; \frac{3k}{2}r$$

Case IV. Let $\binom{k'}{2} - k'^{\frac{9}{8}} \le f(k') < \binom{k'}{2} - \frac{k'}{3}$. Apply R_s, S_t, and $T^{\alpha}_{r,N(r)}$ with parameters:

$$s = \left\lceil \frac{1}{3}k'^{\frac{1}{4}} \right\rceil - k, \qquad t = 1, \qquad \alpha = 1, \qquad r = k' - k - s - \binom{k+s}{2}.$$

Clearly, $s \ge 0$ and we have $k + s + t\binom{k+s}{2} + r = \binom{k+s+1}{2} + r = k'$. Moreover, it is easily seen that $r \ge k'^{\frac{3}{4}}$ for k' large enough, since $r = k' - \binom{k+s+1}{2} \ge k' - \frac{1}{2}(k+s+1)^2 \ge k' - \frac{2}{3}k'^{\frac{1}{2}} \ge k'^{\frac{3}{4}}$. Furthermore, for k large enough (and thus k', r as well), constructibility and exchangeability can be satisfied as can be seen by the following calculations.

- *Constructibility:*

$$\begin{aligned} N(r) &\leq \binom{k'}{2} - \frac{k'}{3} - \binom{k+s+1}{2} r - 2\binom{k+s}{2} \\ &\leq \binom{r}{2} + \frac{1}{4}(k+s+1)^4 - \frac{k'}{3} \;\leq\; \binom{r}{2} + \frac{1}{4}\left(\frac{2}{3}k'^{\frac{1}{4}}\right)^4 - \frac{k'}{3} \\ &\leq \binom{r}{2} - \left(\frac{1}{3} - \frac{4}{81}\right) k' \;\leq\; \binom{r}{2} \end{aligned}$$

- *Exchangeability:* Since $t = 1$, we can iteratively remove vertices in such an order that every removed vertex has maximum degree two when being removed. Thus, a vertex removed from H was incident with at most $r - z + 2$ edges. Assume that every vertex in $A(r, N(r))$ has degree at least $(r-1) - \left(\binom{k+s+1}{2} - 2\right)$ within $A(r, N(r))$. Then a new vertex chosen from $A(r, N(r))$ is incident with at least $(r-z) - \left(\binom{k+s+1}{2} - 2\right) + \binom{k+s+1}{2} = r - z + 2$ edges in H. Therefore we can exchange vertices consecutively such that all vertices from $A(r, N(r))$ are chosen. The minimum degree of a vertex in $A(r, N(r))$ is $\lfloor \frac{2N(r)}{r} \rfloor$. Thus, we have to prove $\left\lfloor \frac{2N(r)}{r} \right\rfloor \geq (r-1) - \left(\binom{k+s+1}{2} - 2\right)$ what is equivalent to $2N(r) \geq 2\binom{r}{2} - \binom{k+s+1}{2} r + 2r$ since k, r and s are natural numbers. The inequality can be seen as follows:

$$\begin{aligned} 2N(r) &\geq 2\left(\binom{k'}{2} - k'^{\frac{9}{8}}\right) - 2\binom{k+s+1}{2} r - 4\binom{k+s}{2} \\ &= 2\binom{r}{2} + 2\binom{\binom{k+s+1}{2}}{2} - 2k'^{\frac{9}{8}} - 4\binom{k+s}{2} \end{aligned}$$

Finally, we obtain the desired statement by the following calculations:

$$\begin{aligned} &2\binom{\binom{k+s+1}{2}}{2} + \binom{k+s+1}{2} r - 2r - 2k'^{\frac{9}{8}} - 4\binom{k+s}{2} \\ &\quad\geq \frac{1}{4}(k+s)^4 + (k+s)^2\left(\frac{1}{2}r - 2\right) - 2r - 2k'^{\frac{9}{8}} \;\geq\; \frac{1}{9}k'^{\frac{1}{2}}\left(\frac{1}{2}k'^{\frac{3}{4}} - 2\right) - 4k'^{\frac{9}{8}} \\ &\quad\geq \frac{1}{18}k'^{\frac{5}{4}} - 5k'^{\frac{9}{8}} \;\geq\; 0 \end{aligned}$$

Case V. Let $\binom{k'}{2} - \frac{k'}{3} \leq f(k') \leq \binom{k'}{2}$. Apply R_s, S_t, and $T^{\alpha}_{r,N(r)}$ with parameters:

$$s = 0, \qquad t = 0, \qquad \alpha = 1, \qquad r = k' - k.$$

Clearly, we have $k + s + t\binom{k+s}{2} + r = k + r = k'$. For k large enough (and thus k', r as well), constructibility and exchangeability can be satisfied as can be seen by the following arguments.

- *Constructibility:*

$$N(r) \;\leq\; \binom{k+r}{2} - kr - \binom{k}{2} \;=\; \binom{r}{2}$$

- *Exchangeability:* Let B be the densest k-vertex subgraph of H. Assume that G has no clique of size k (which in fact, is the only interesting case to consider). Hence, B is not a clique. Since B is the densest subgraph, each vertex of H which does not belong to B is adjacent to at most $k-1$ vertices of B. Thus, on the one hand, removing all vertices in $H \setminus B$ yields a loss of at most $z(r-z) + \binom{z}{2} + z(k-1)$ edges. On the other hand, since $A(r, N(r))$ misses at most $\frac{k'}{3} \leq \frac{r}{2}$ edges to be complete, each vertex of the quasi-regular graph $A(r, N(r))$ is adjacent to at least $r-2$ vertices, thus not connected to at most one vertex other than itself. Consequently, choosing all z not-yet-chosen vertices of $A(r, N(r))$ adds at least $(r-z)z + \binom{z}{2} - z + zk = z(r-z) + \binom{z}{2} + z(k-1)$ edges. Thus, an exchange of vertices is possible without loosing edges in number.

❑

Now we are able to formulate the result of Theorem 9 in terms of γ-Cluster.

Corollary 10. *Let $\gamma = 2+\Omega(\frac{1}{k^{1-\varepsilon}})$ for some $\varepsilon > 0$ be polynomial-time computable, $\gamma(k) \leq k-1$. Then γ-Cluster is* **NP***-complete.*

5 Conclusion

In this paper we have proved that density-based clustering in graphs is inherently hard. The main result states that finding a k-vertex subgraph with average degree at least $\gamma(k)$ is **NP**-complete if $\gamma = 2 + \Omega(\frac{1}{k^{1-\varepsilon}})$ and solvable in polynomial time if $\gamma = 2 + O(\frac{1}{k})$. In particular, for constant average-degree that means that detecting whether there is a k-vertex subgraph with average degree at least two is easy but with average degree at least $2+\varepsilon$ it is intractable. Since the **NP**-threshold is so tremendously low, it seems inevitable to explore how the problem behaves in relevant graph-classes, e.g., in sparse graphs or graphs with small diameter. Sparsity, however, is not expected to lift the **NP**-threshold.

Though detecting a subgraph of a certain size and a certain density is an important algorithmic issue, the original problem intended to be solved is Maximum γ-Cluster: compute the largest subgraph with average degree over some γ-value. Of course, this problem is intimately related to γ-Cluster, and in fact, we have the same tractable-intractable threshold as for the decision problem. The main open question is: how good is Maximum γ-Cluster approximable depending on γ? For instance, for $\gamma(k) = k-1$ (i.e., Maximum Clique), it is known to be approximable within $O\big(\frac{n}{(\log n)^2}\big)$ [5] but not approximable within $n^{\frac{1}{2}-\varepsilon}$ unless $\mathbf{P} = \mathbf{NP}$ [14]. How do these results translate to intermediate densities?

Acknowledgment. For helpful hints and discussions we are grateful to Christian Glaßer, Ernst W. Mayr, and Alexander Offtermatt-Souza. We also thank Yuichi Asahiro, Refael Hassin, and Kazuo Iwama for providing us with an early draft of [3].

References

[1] C. J. Alpert and A. B. Kahng. Recent developments in netlist partitioning: A survey. *Integration: The VLSI Journal*, 19(1-2):1–81, 1995.

[2] S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. *Journal of Computer and System Sciences*, 58(1):193–210, 1999.

[3] Y. Asahiro, R. Hassin, and K. Iwama. Complexity of finding dense subgraphs. *Discrete Applied Mathematics*, 121(1-3):15–26, 2002.

[4] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *Journal of Algorithms*, 34(2):203–221, 2000.

[5] R. Boppana and M. M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT*, 32(2):180–196, 1992.

[6] M. S. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Proceedings 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization*, volume 1913 of *Lecture Notes in Computer Science*, pages 84–95. Springer-Verlag, Berlin, 2000.

[7] J. Cong and M. Smith. A parallel bottom-up clustering algorithm with applications to circuit partitioning in VLSI design. In *Proceedings 30th ACM/IEEE Design Automation Conference*, pages 755–760. ACM Press, New York, 1993.

[8] A. Czygrinow. Maximum dispersion problem in dense graphs. *Operations Research Letters*, 27(5):223–227, 2000.

[9] U. Faigle and W. Kern. Computational complexity of some maximum average weight problems with precedence constraints. *Operations Research*, 42(4):1268–1272, 1994.

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

[11] U. Feige and M. Seltser. On the densest k-subgraph problem. Technical Report CS97-16, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot, Israel, 1997.

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

[13] O. Goldschmidt, D. Nehme, and G. Yu. On the set union knapsack problem. *Naval Research Logistics*, 41(6):833–842, 1994.

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

[15] D. J.-H. Huang and A. B. Kahng. When cluster meet partitions: New density-based methods for circuit decomposition. In *Proceedings European Design and Test Conference*, pages 60–64. IEEE Computer Society Press, Los Alamitos, 1995.

[16] J. M. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins. The Web as a graph: measurements, models, and methods. In *Proceedings 5th International Conference on Computing and Combinatorics*, volume 1627 of *Lecture Notes in Computer Science*, pages 1–17. Springer-Verlag, Berlin, 1999.

[17] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins. Trawling the Web for emerging cyber-communities. *Computer Networks*, 31(3):1481–1493, 1999.

[18] T. Murata. Discovery of Web communities based on the co-occurrence of references. In *Proceedings 3th International Conference on Discovery Science*, volume 1967 of *Lecture Notes in Artificial Intelligence*, pages 65–75. Springer-Verlag, Berlin, 2000.

[19] D. Nehme and G. Yu. The cardinality and precedence constrained maximum value sub-hypergraph problem and its applications. *Discrete Applied Mathematics*, 74(1):57–68, 1997.

[20] A. Srivastav and K. Wolf. Finding dense subgraphs with semidefinite programming. In *Proceedings International Workshop on Approximation Algorithms for Combinatorial Optimization*, volume 1444 of *Lecture Notes in Computer Science*, pages 181–191. Springer-Verlag, Berlin, 1998.

[21] P. Turán. On an extremal problem in graph theory. *Matematikai és Fizikai Lapok*, 48:436–452, 1941. In Hungarian.