algorithms

MDPI

Article

In Search of the Densest Subgraph

András Faragó and Zohre R. Mojaveri *

Department of Computer Science, Erik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas, P.O.B. 830688, MS-EC31, Richardson, TX 75080, USA

* Correspondence: zohre.r.mojaveri@utdallas.edu

Received: 23 June 2019; Accepted: 30 July 2019; Published: 2 August 2019

Abstract: In this survey paper, we review various concepts of graph density, as well as associated theorems and algorithms. Our goal is motivated by the fact that, in many applications, it is a key algorithmic task to extract a densest subgraph from an input graph, according to some appropriate definition of graph density. While this problem has been the subject of active research for over half of a century, with many proposed variants and solutions, new results still continuously emerge in the literature. This shows both the importance and the richness of the subject. We also identify some interesting open problems in the field.

Keywords: dense subgraph; algorithms; graph density; clusters in graphs; big data

1. Introduction and Motivation

In the era of big data, graph-based representations became highly popular to model various real-world systems, as well as diverse types of knowledge and data, due to the simplicity and visually appealing nature of graph models. The specific task of extracting a *dense subgraph* from a large graph has received a lot of attention, since it has direct applications in many fields.

While the era of big data is still relatively young, the study and application of dense subgraphs started much earlier. The likely first application was in sociology, for analyzing the cohesive structure of social groups. The most obvious cohesive structure is represented by a complete graph (clique), which was formally used in a sociological study as early as 1949 [1].

In this paper, we are mostly interested in the algorithmic complexity of dense subgraph extraction. A natural first question, with major impact on the algorithmic hardness of the task, is regarding how to meaningfully define density: Which subgraphs of a large graph can be reasonably considered dense? We analyze the issue in great detail in the next section. To motivate it, however, let us first list a number of applications (for a collection of references, see, e.g., Gionis and Tsourakakis [2]).

- In a social network, such as Facebook, a typical graph representation has nodes corresponding to individuals, and the edges capture some relation or interaction between them, e.g., friendship. In this model, a dense subgraph represents a community. Many types of communities exist, as the tendency of people with similar tastes, choices, and preferences to get associated may lead to the formation of communities.
- In a communication network, a dense subgraph can capture a congested part of the network, assuming that the edges correspond to those links that have traffic load over some threshold. Identifying such congested parts, and then taking appropriate action to relieve the congestion, can have a major impact on network performance.
- In a mobile ad hoc radio network, a dense subgraph can represent a subnetwork with high radio interference. These parts tend to offer lower throughput, thus identifying them can be useful to avoid degraded performance.

- In the World Wide Web (WWW), a natural graph representation has the websites as its nodes, and the hyperlinks between them as edges. A densely interconnected part may indicate a web community, such as a group of content creators sharing a common interest.
- Another application in the WWW is the detection of *link spam*. Link spam is the posting of out-of-context links on websites, discussion forums, blog comments, guest-books or other online venues that display user comments. The purpose of link spam is to increase the number of external links pointing to a page the spammer wants to promote, with the goal of increasing page rank and improving its position in search engine results. Then, the higher rankings in web searches lead to greater visibility over competitors, more visitors and potentially more paying customers.
- In bioinformatics, dense subgraphs are used for finding molecular complexes in protein–protein interaction networks, for discovering regulatory motifs in genomic DNA, and finding complex patterns in the gene annotation graph.
- In the field of finance, dense subgraphs have been used for discovering migration patterns in financial markets.
- In data mining, *correlation mining* represents observation sequences as graph nodes, and strong correlation between them as edges. Then, a dense subgraph can capture highly correlated groups of entities, which has been used in stock market analysis, and computational biology.
- Other use cases of dense subgraphs include graph compression, graph visualization, clustering, real-time identification of important stories on Twitter, and many other data mining and knowledge discovery applications.

2. Preliminaries

Let us summarize some standard notations that are used throughout the paper.

If G is a graph, then $V(G)$ is its vertex (node) set, and $E(G)$ is its edge set. We restrict ourselves to undirected simple graphs, where simple means that there are no self-loops and parallel edges. The subgraph that is sought in an input graph G is denoted by S, with vertex and edge sets $V(S)$ and $E(S)$, respectively.

We use the notation $d(x)$ for the degree of a vertex x. If it is not obvious from the context in which graph the degree is measured, then we put the name of the graph in the index. For example, if S is a subgraph of G, then it is possible that $d_S(x) \neq d_G(x)$. It always holds, however, that $d_S(x) \leq d_G(x)$. The average degree in a subgraph S is denoted by $\overline{d}(S)$, defined as

$$\overline{d}(S) = \frac{\sum_{x \in V(S)} d_S(x)}{|V(S)|}.$$

Various density measures are denoted by $\rho_i(S)$, where the index i distinguishes the type.

3. Measures of Graph Density and Associated Results

In this section, we review various concepts of graph density, and corresponding results about the problem of finding the densest subgraph, according to the considered density type. Our intent is to show some typical and important cases; we do not aim at the (almost impossible) goal of providing a completely exhaustive list.

We divide the density measures into two major groups: (1) cases in which the densest subgraph can be found in polynomial time; and (2) cases where the problem is NP-hard. Among the open problems, we also mention some density types for which it is not known to which group they belong.

3.1. Polynomial-Time Solvable Cases

3.1.1. Edge Density: Subgraph with Maximum Average Degree

A very natural first measure of density is the number of edges divided by the number of nodes in the subgraph, since more edges on the same number of nodes provide an intuitively denser, more cohesive subgraph. Denoting this density by ρ_1, we get

$$\rho_1(S) = \frac{|E(S)|}{|V(S)|}.$$

This is sometimes called *edge density*, or just density, for short. Observe that $|E(S)|$ is precisely half of the sum of the degrees in the subgraph S, since each edge contributes 2 to the sum of degrees. Therefore, we can write

$$\rho_1(S) = \frac{1}{2}\frac{\sum_{x \in V(S)} d_S(x)}{|V(S)|} = \frac{1}{2}\,\bar{d}(S),$$

that is, $\rho_1(S)$ is half of the average vertex degree in S. Thus, finding a subgraph S that maximizes the density $\rho_1(S)$ is equivalent to finding a subgraph with maximum average degree.

This task, while not at all trivial, can be solved in polynomial time, in several different ways. A network flow based solution was already published in 1982 by Picard and Queyranne [3]. The running time was improved by Gallo, Grigoriadis and Tarjan [4] in 1989, keeping the network flow based approach. Charikar [5] showed in 2000 that the density value $\rho_1(S)$ of the densest subgraph S is equal to the optimal solution of the following simple linear program: assign a variable x_{ij} to every edge (i, j) and a variable y_i to every node i, and consider the linear program

$$\max \sum_{i,j} x_{ij}$$

Subject to

$$\forall i, j: \; x_{ij} \leq y_i$$

$$\forall i, j: \; x_{ij} \leq y_j$$

$$\sum_i y_i \leq 1$$

$$\forall i, j: \; x_{ij}, y_i \geq 0.$$

From this not only a polynomial-time solution can be derived, but it also allows a generalization to directed graphs. Furthermore, it also gives rise to a very simple 2-approximation algorithm. Specifically, the approximation algorithm starts with $S_0 = G$, and then removes a vertex $x \in V(S_0)$ for which $d_{S_0}(x)$ is minimum, i.e., a minimum degree vertex from S_0, to obtain a new (smaller) subgraph S_1. Then, again, a minimum degree vertex is removed from S_1, to obtain S_2, and so on, until we arrive at an empty graph, after at most $n = |V(G)|$ iterations. Then, from the generated graph sequence $S_0, S_1, S_2, \ldots$, we select the one for which $\rho_1(S_i)$ is the largest. Charikar [5] proved that this surprisingly simple algorithm guarantees an approximation ratio not worse than 2, that is, it finds a subgraph with at least half of the optimum density.

Another polynomial-time algorithm was given by Dong and Liu [6] in 2004. In 2008, Faragó [7] provided an algorithm that reduces the problem to bipartite matching, also allowing generalizations to more complex density concepts (see Section 3.1.4). Apparently, the fastest asymptotic running time to date for finding the exact optimum is achieved by the parametric maximum flow based algorithm of Gallo, Grigoriadis and Tarjan [4], which runs in time $O(nm \log(n^2/m))$ on a graph of n vertices and m edges.

3.1.2. k-Core: Subgraph with Largest Minimum Degree

Another reasonable density concept is obtained if we use the *minimum degree* of the subgraph, rather than its average degree. In some sense, this provides a worst-case guarantee for the subgraph density, rather than an average-case guarantee.

Definition 1. *The k-core of a graph is the largest subgraph in which every vertex has degree at least k. The largest value of k for which the graph has a non-empty k-core is called the* degeneracy *of the graph.*

The concept was originally introduced by Lick and White [8] in 1970, and then analyzed in many other papers. The k-core of a graph is uniquely determined for every k, due to the fact that the property of having minimum degree at least k is closed with respect to the union of subgraphs. For different values of k, the k-cores form a nested structure.

A great advantage of the k-core is that it is very easy to find algorithmically. The principle of the algorithm can be described in one sentence: delete all nodes with degree $< k$, and repeat this in the remaining graph, until either all nodes have degree $\geq k$, or the remaining graph is empty. (In the latter case, we usually say that there is no k-core for the given value of k, rather than saying that the k-core is empty). Note that the repetition is needed because the removal of nodes may decrease some degrees in the remaining graph. With a little further sophistication, this algorithm can be implemented in linear time, using the so-called bucket queue data structure (see Matula and Beck [9]).

The simplicity of finding the optimal k-core leads to the question: Why do we need any other, more complicated, density concepts at all, if the k-core is so easy to find? A possible answer is that the k-core can behave counter-intuitively, with respect to the intuitive concept of the edge density, introduced in Section 3.1.1. Specifically, Tatti and Gionis [10] demonstrated that the k-core can have lower edge density than an ℓ-core with $\ell < k$ in the same graph, even though we intuitively expect that a core with higher k value should be more dense.

3.1.3. Subgraph with Maximum k-Clique Density

A generalization of the edge density (see Section 3.1.1) was introduced by Tsourakakis [11]. It is called k-clique density, and is defined as the number of k-cliques in the subgraph S, divided by $|V(S)|$. The classic edge density is the special case with $k = 2$. Another important special case is $k = 3$, which is called *triangle density*.

A key result in [11] is that for any constant integer $k \geq 2$ the subgraph with the highest k-clique density can be found in polynomial time. The practical significance of the k-clique density is that it provides a better approximation for the maximum clique, which is **NP**-hard to find (see Section 3.2.1). According to numerical experiments [11], already the case of triangle density ($k = 3$) leads to large near-cliques in real-world graphs.

3.1.4. Subgraph with Maximum F-Density

A further generalization was introduced by Faragó [7]. For notational convenience, if $\mathbf{F}$ is a set of graphs, then we say that a graph G is an $\mathbf{F}$-graph if $G \in \mathbf{F}$.

Definition 2. *Let $\mathbf{F}$ be a fixed finite set of graphs. The $\mathbf{F}$-density of a graph S is defined by*

$$\rho(S, \mathbf{F}) = \frac{M(S, \mathbf{F})}{|V(S)|}$$

where $M(S, \mathbf{F})$ denotes the number of $\mathbf{F}$-graphs in S.

Let us illustrate the concept by some simple examples.

Example 1. *If* **F** *consists of only one graph, which is a single edge, then we get back the definition of the edge-density of Section 3.1.1:*

$$\rho_1(S) = \frac{|E(S)|}{|V(S)|} = \frac{1}{2}\frac{\sum_x d_S(x)}{|V(S)|}.$$

Example 2. *If* **F** *consists of a triangle (a complete graph on three vertices), then we obtain the* triangle-density*: the number of triangles divided by the number of vertices, which we have already encountered in Section 3.1.3. Similarly, if* **F** *consists of a k-clique, then we get back the k-clique density of Section 3.1.3.*

Example 3. *Let* **F** *consist of a single graph that is a path of length 2 (on three vertices). Then, each occurrence of such a path in a graph can be described by the position of the middle vertex of the path and the two edges adjacent to it. It is easy to see that the number of such paths with middle vertex $x \in V(S)$ is $\binom{d_S(x)}{2}$, and if two paths have different middle vertices then they cannot coincide. Therefore, the resulting density measure is*

$$\rho_2(G) = \frac{1}{|V(S)|}\sum_x \binom{d_S(x)}{2}.$$

Example 4. *Let us combine Examples 1 and 3, and put now two graphs in* **F***: a single edge and a path of length 2. It yields the density*

$$\rho_3(S) = \frac{\sum_x \binom{d_S(x)}{2} + \frac{1}{2}\sum_x d_S(x)}{|V(S)|} = \frac{1}{2}\frac{\sum_x d_S^2(x)}{|V(S)|}.$$

That is, instead of the average degree, we aim here at maximizing the average of the squared *degrees. The effect is that larger degrees have bigger contribution, so the subgraph will be pulled towards larger degrees.*

A key result of the paper Faragó [7] is that for any fixed finite set **F** of graphs, and for any input graph G, a subgraph with maximum **F**-density can be found in polynomial time. This allows efficiently finding maximally dense subgraphs with respect to new density measures. For example, Examples 3 and 4 provide densities that differ non-trivially from all cases covered by the previous sections. On the other hand, if **F** is not fixed, then the problem becomes **NP**-hard.

Let us mention incidentally that the optimization of the **F**-density in [7] is based on another graph optimization problem that is called the maximization of the weighted *independence ratio*, which is quite interesting on its own. To explain it, let us introduce some notations. A subset of vertices is called *independent* if there is no edge between any two of the nodes in the subset. Let $\mathcal{F}(G)$ be the family of all nonempty independent sets in the graph G. For a set A of vertices, $\Gamma(A)$ denotes the set of neighbors of A, that is,

$$\Gamma(A) = \{y|\ \exists x \in A:\ (x,y) \in E(G)\}.$$

Assume that a positive integer weight function w is given on the nodes. The weight of a set A is denoted by $w(A)$ and is defined as the sum of the weights of vertices in A.

Definition 3 (Independence ratio). *The* independence ratio *of an independent set $A \in \mathcal{F}(G)$ with $\Gamma(A) \neq \emptyset$ is defined as*

$$\gamma^{(w)}(A) = \frac{w(A)}{w(\Gamma(A))}.$$

If $\Gamma(A) = \emptyset$, then we define $\gamma^{(w)}(A) = \infty$.

We are looking for an independent set with *maximum independence ratio*. It turns out that an independent set of maximum independence ratio can be found in polynomial time when the task is restricted to the class of *slender graphs* (defined below), while the problem remains **NP**-hard for the general case.

Definition 4 (Slender graph). *A graph is called* slender, *under the weighting w, if there exists an independent set A in the graph with $w(A) \geq w(\Gamma(A))$.*

Now, the result about the independence ratio, proved in [7], is this:

Theorem 1 (Complexity of independence ratio). *Let w be an arbitrary positive integer weighting function. An independent set with maximum independence ratio can be found in polynomial time in the class of graphs that are slender under the weighting w. Without the slenderness restriction the task is* **NP***-hard, even for the unweighted ($w \equiv 1$) case.*

3.1.5. Densest Subgraph with Concave Size Function

A different generalization of the edge density was proposed by Kawase and Miyauchi [12]. Let f be a positive real-valued, non-decreasing function. The f-density of a graph S is defined by

$$\rho_4(S, f) = \frac{|E(S)|}{f(|V(S)|)}.$$

They also allow that the edges are assigned non-negative weights, and then the density becomes

$$\frac{w(S)}{f(|V(S)|)},$$

where $w(S)$ denotes the sum of edge weights in S.

Consider the case where f is a concave function, e.g., $f(|V(S)|) = \sqrt{|V(S)|}$. Then, $f(|V(S)|)$ grows slower than linear. This means the density gives preference to larger vertex sets. For example, if two graphs S_1, S_2 have $m_1 = 100$ and $m_2 = 200$ edges, and $n_1 = 25$ and $n_2 = 50$ vertices, respectively, then their edge density is the same: $\rho_1(S_1) = \rho_1(S_2) = 100/25 = 200/50 = 4$. However, with $f(|V(S)|) = \sqrt{|V(S)|}$, we get $\rho_4(S_1, f) = \frac{100}{\sqrt{25}} = 20$ and $\rho_4(S_2, f) = \frac{200}{\sqrt{50}} = \frac{2}{\sqrt{2}} \cdot \frac{100}{\sqrt{25}} = \sqrt{2} \cdot 20 > \rho_4(S_1)$. Thus, indeed, the larger of the equally dense graphs is preferred.

An important general result of Kawase and Miyauchi [12] is that the f-densest subgraph can be found in polynomial time for any real-valued, non-decreasing, polynomial-time computable concave function. On the other hand, if the function is strictly convex, then the problem becomes **NP**-hard. Nevertheless, approximate optimization is still possible in the convex case, with an approximation factor depending on the function. For example, if f is the quadratic function $f(|V(S)|) = \lambda|V(S)| + (1-\lambda)|V(S)|^2$, with a constant $\lambda \in [0, 1)$, then a constant approximation ratio of $(2-\lambda)/(1-\lambda)$ can be achieved in polynomial time. All these results carry over to the edge weighted case.

3.1.6. Densest Subgraph with Specified Subset

Motivated by gene annotation graphs in computational biology, Saha et al. [13] introduced the problem of finding a maximum edge density subgraph, under the constraint that it contains a specified subset of nodes. While this constraint makes the problem harder, it remains solvable in polynomial time. Saha et al. [13] gave an $O(n^4 \log n)$-time algorithm for the problem in graphs with n vertices, using network flows. Later, Chen et al. [14] found an $O(n^2)$-time greedy approximation algorithm with approximation ratio $2(1 + \frac{k}{3})$, for the case where the subgraph is required to contain at least k vertices. The results carry over to the edge weighted case, as well.

3.1.7. Dense Subgraph with Sparse Cut

Miyauchi and Kakimura [15] defined a density measure, which aims at finding a subgraph S, such that S is dense internally, but it is sparsely connected to the rest of the graph. In this sense, it kind of "bumps out" of the graph. Therefore, it may better capture a community in a social network, than an approach that focuses purely on the internal density.

To implement the objective, they take two types of edges into account: internal and cut edges. The internal edges are the ones within the subgraph S, while the cut edges are those that have precisely one endpoint in S. With non-negative edge weights, let $w_{in}(S)$ be the total weight of internal edges, and $w_{cut}(S)$ be the total weight of cut edges. With an arbitrary non-negative parameter α, they defined the following density measure:

$$f_\alpha(S) = \frac{w_{in}(S) - \alpha w_{cut}(S)}{|V(S)|}.$$

The role of the parameter α is to allow controlling the relative importance of the two aspects: internal density vs. sparse cut to the rest of the graph.

The main result of Miyauchi and Kakimura [15] is that a subgraph S that maximizes $f_\alpha(S)$ can be found in polynomial time for any α. It is particularly interesting that some closely related problems are **NP**-hard. An example is the *sparsest cut* problem, which aims at minimizing the value of $w_{cut}(S)/|V(S)|$ for $|V(S)| \leq |V(G)|/2$. Note that, if α is so large that $w_{in}(S) \ll \alpha w_{cut}(S)$, then, with the maximizing S, the value of $-f_\alpha(S)/\alpha$ is close to the sparsest cut value.

The authors also presented a nearly linear time approximation algorithm for the problem.

3.1.8. Subgraph with Maximum Edge-Connectivity

The *edge-connectivity* of a graph is the minimum number of edges that need to be deleted in order to disconnect it, it is denoted by $\lambda(G)$. Since we only deal here with edge-connectivity (and not vertex connectivity), we call it simply *connectivity.* We call a graph k-connected if its connectivity is k, i.e., $\lambda(G) = k$. This also means the graph has a *cut* (a set of disconnecting edges) of size k, but no smaller cut. There are many results about graph connectivity. In particular, it can be computed in polynomial time, and a corresponding minimum cut can also be found in polynomial time (see, e.g., Nagamochi and Ibaraki [16]).

Here, we only focus on connectivity as a density measure. It is intuitive that a graph with high connectivity cannot be too sparse. For example, if it is k-connected, then each degree must be at least k, since otherwise a node could be separated from the rest by less than k edges. The converse, however, is not true: if every degree is $\geq k$, it alone may not make the graph k-connected (not even connected).

It is reasonable to capture a highly connected community by a subgraph with high connectivity. Therefore, we consider the task of finding a subgraph that has the highest possible connectivity among all subgraphs of the given input graph. This and related questions were investigated by Matula [17]. Here, we outline a simple algorithm to find a subgraph with the highest connectivity, assuming that a subroutine is available to find a minimum cut in any given graph. It is interesting that, while it may first appear hard to find the most connected subgraph among exponentially many subgraphs, the problem actually has a simple polynomial-time algorithm, once we can use a minimum cut finding subroutine.

Principle of the Algorithm

- Find a minimum cut in the input graph G; let its size be denoted by $k = \lambda(G)$.
- Let A, B be the two node sets into which the minimum cut divides the graph. A key observation is that, if there is any subgraph G_0 with $\lambda(G_0) > k$, then it must be either fully in A or fully in B. This is because, if G_0 had nodes on both sides, then the found cut of size k would separate these nodes, too. G_0, however, cannot be separated by k edges, due to $\lambda(G_0) > k$. Thus, if such a G_0 exists, then it must be either fully in A or fully in B.
- Let G_1, G_2 denote the most connected subgraphs within the sets A, B, respectively. We can find them by recursively calling the algorithm for the smaller graphs induced by A and B.
- Return the graph that has the highest connectivity among the three graphs G, G_1, and G_2.

Remark 1. *The algorithm can be easily modified such that it finds a subgraph that has the maximum number of nodes among the possibly multiple subgraphs of maximum connectivity.*

Analysis: At most how many minimum cut computations are needed?

At first, it may not be obvious that the algorithm calls the minimum cut subroutine only polynomially many times. After all, the graph is repeatedly split into two parts by a minimum cut computation, and then the algorithm is recursively called for each part. This may give the impression that in the worst case we may make an exponential number of minimum cut computations, since we keep splitting each graph, and, therefore, keep doubling the number parts. This impression, however, is incorrect. Let $f(n)$ denote the number of minimum cut computations we carry out when the recursive algorithm is run on an n-node graph.

Claim: $f(n) \leq n - 1$.

Proof. We use induction. The base case $n = 1$ is trivial, since $f(1) = 0$, as no minimum cut computation is needed for $n = 1$. In the recursive step we partition the graph into two parts, let the number of nodes in them be k and $n - k$. We do not know the value of k in advance, only that it is an integer between 1 and $n - 1$. Nevertheless, by the induction hypothesis, we can assume $f(k) \leq k - 1$ and $f(n - k) \leq n - k - 1$. The total number of minimum cut computations we make for n nodes is at most the sum of $f(k)$ and $f(n - k)$, plus one more to generate the split. Therefore, $f(n) \leq f(k) + f(n - k) + 1 \leq (k - 1) + (n - k - 1) + 1 = n - 1$. Observe that this holds regardless of the value of k, thus it indeed completes the inductive proof. □

3.2. NP-Hard Density Measures and Related Results

3.2.1. Maximum Clique

The archetypal example of a dense subgraph is a *clique*, i.e. a complete subgraph. Of course, we do not just look for *any* clique, since that would be trivial (a single edge is already a clique). Normally, we want to find a *maximum clique*, that is, the one with the largest possible number of vertices. This task, however, is one of the best known and earliest **NP**-complete problems (see, e.g. Garey and Johnson [18]). Specifically, the following problem is **NP**-complete:

Problem: CLIQUE
Input: Graph G, positive integer k
Question: Does G have a clique of size at least k?

The above is the *decision version* of the problem. The corresponding *optimization version* or *search problem* is **NP**-hard:

Problem: MAXCLIQUE
Input: Graph G
Task: Find a maximum clique in G.

At this point, one may wonder: If we know the answer to the decision problem for the input graph G, could it make the optimization/search easier? The answer is no (see, e.g., Kosub [19]):

Theorem 2. *If* **P** $\neq$ **NP**, *then there is no polynomial-time algorithm to find a clique of size k in a graph G, even if it is guaranteed that G has a clique of size k.*

On the other hand, if a subroutine (often referred to as *oracle*) is available that can solve the decision problem for *any* graph, not just for the input at hand, then calling it only polynomially many times is already enough to solve the optimization/search problem efficiently, assuming that an oracle call is counted as a single step. The reason is that one can reduce a large instance to a smaller one, and repeating it recursively step by step allows recovering the nodes of a maximum clique. This kind of approach applies to every **NP**-complete problem, and it is called *self-reducibility* [18]. Metaphorically speaking, if we know that the haystack contains a needle, but this fact alone does not make it easier to find the needle. If, however, we can decide for *any* haystack whether it contains a needle, that already allows finding the needle efficiently.

Faster Exponential-time Algorithms

Even though no polynomial-time algorithm is expected to exist to find a maximum clique, this does not mean that the best possibility is exhaustive search. Naive exhaustive search of all node subsets in a graph with n nodes would take examining 2^n subsets in the worst case. However, significantly faster, yet still exponential-time, algorithms exist. Robson [20] published an algorithm that runs in $O(1.2108^n)$ time. It was further improved by Robson in a technical report [21], to $O(1.1888^n)$ time, which is apparently the fastest to date. Note that there can be a dramatic difference between 2^n and 1.1888^n. For example, in a small graph with only $n = 20$, we can already see a big difference: 2^n is more than a million, while 1.1888^n is less than 32.

Approximation

Since there is no realistic chance to always find a truly maximum clique in any input graph faster than exponential time, it is natural to look for approximations. However, it is far from easy. A general result is due to Boppana and Haldórsson [22]; they provided a polynomial-time algorithm that guarantees an approximation ratio of $O(n/\log^2 n)$ for any graph of n vertices. The approximation ratio was later improved by Feige [23] by a factor of $O((\log\log n)^2/\log n)$ to $O(n(\log\log n)^2/\log^3 n)$.

To put the results in context, note that an $O(n)$ approximation is trivial, as a single node already achieves it. The above factors are slightly better than $O(n)$, but not much. They cannot even shave off an $O(n^\epsilon)$ factor for any $\epsilon > 0$, despite their rather sophisticated methods.

Inapproximability

The fact that decades of research could not turn up a polynomial-time approximation algorithm for MAXCLIQUE with approximation ratio $O(n^{1-\epsilon})$ for any $\epsilon > 0$, no matter how small, seems to indicate that such an algorithm perhaps does not exist at all, under some plausible complexity theoretic assumptions. After a number of gradually improving earlier results, eventually it was proved by Zuckerman [24] that this is indeed the case, and the only assumption needed is **P** $\neq$ **NP**. Specifically, he proved that, for any $\epsilon > 0$, finding an $O(n^{1-\epsilon})$ approximation of the maximum clique in an n-vertex graph is **NP**-hard.

The above result means that MAXCLIQUE indeed behaves very rigidly regarding approximation: while an $O(n)$ approximation is trivial, anything of the form $O(n^\alpha)$ with $\alpha < 1$ is virtually impossible, i.e., no $O(n^{1-\epsilon})$ approximation exists for any $\epsilon > 0$, unless **P** = **NP**.

Special Graph Classes

While the above results show that in general graphs there is no chance to efficiently find a maximum clique, or even a reasonably good approximation, it does not mean that one cannot achieve success when the input is restricted to some special graph class. Examples of well-known classes with polynomial-time MAXCLIQUE algorithms are perfect graphs, planar graphs, unit disc graphs (with a given geometric representation), various types of intersection graphs, etc. If one also includes less well known classes, then actually there are hundreds of classes, for which MAXCLIQUE can be solved in polynomial time (see [25]).

Listing All Maximal Cliques

While finding a maximum clique is hard, listing *all* inclusion-wise maximal cliques, which, of course, include all maximum (notice the difference in word usage: maximum refers to the largest, while maximal refers to inclusion, meaning that it cannot be extended into a larger clique, but it may not itself be largest) cliques also can be done in polynomial time. This may first sound contradictory. However, there are usually exponentially many maximal cliques, and they all have to be listed in the output, which allows us exponential running time in terms of the graph size. To have some

efficiency, we expect that, on average, the algorithm does only a polynomial amount of work per output configuration.

However, regarding such enumeration algorithms, a stricter quantity that is often considered as a measure of efficiency is called *delay*. We say that an enumeration algorithm has *polynomial delay*, if it generates the configurations (in our case the maximal cliques), one after the other in some order, in such a way that the delay until the first output, the delay between any two consecutive outputs, and the delay until it stops after the last output are bounded by a polynomial in the input size (the graph size).

Despite the hardness of MAXCLIQUE, listing all maximal cliques with polynomial delay is possible. Such an algorithm was developed by Tsukiyama, Ide, Ariyoshi, and Shirakawa [26]:

Theorem 3. *There is an algorithm enumerating all maximal cliques of a graph with $O(n^3)$ delay, using $O(n+m)$ space, where n is the number of vertices and m is the number of edges.*

What if we want to list all maximal cliques in some specific order, not just any order? For certain orders, such as lexicographic order, this is also possible with polynomial delay, but needs further tricks. Johnson, Papadimitriou, and Yannakakis [27] proved the following result:

Theorem 4. *There is an algorithm enumerating all maximal cliques of an n-vertex graph in lexicographic order with $O(n^3)$ delay.*

They also proved that the same is likely impossible for some other orders:

Theorem 5. *If* **P** $\neq$ **NP**, *then there is no algorithm that generates for any given graph all maximal cliques in inverse lexicographic order with polynomial delay.*

3.2.2. Constrained Dense Subgraphs and Quasi-Cliques

In view of the hardness of finding (or even approximating) a maximum clique, it is not surprising that researchers were looking for various density concepts that lead to subgraphs not too far from a large clique. Below, we review some of the results in this direction.

Plexes

A graph that is close to a clique in the sense that only a few edges are missing at every vertex to connect it to all others, is called a *plex*. The formal definition is this:

Definition 5. *A k-plex in a graph G is a subgraph S, such that for every $x \in V(S)$ it holds that $d_S(x) \geq |V(S)| - k$.*

With this definition, a clique is a 1-plex, since every node has degree $|V(S)| - 1$. For $k > 1$, we allow that some edges are missing from the clique. However, for moderate values of k, not too many edges are missing, so the graph is close to a clique.

As plexes approximate cliques rather closely, it is not surprising that finding them is hard. It is expressed by the following theorem (see Kosub [19]):

Theorem 6. *For any positive integer k, it is* **NP***-complete to decide whether a given graph contains a k-plex with at least a given number of nodes.*

An interesting feature of k-plexes is that one can prove a relationship between the value of k, and the *diameter* of the graph. The diameter $diam(G)$ of a connected graph G is the largest hop-distance that can occur between any two nodes. One can prove the following (see Kosub [19]):

Theorem 7. *Let G be a connected n-vertex k-plex, with $1 \leq k \leq n$. Then,*

- $k < \frac{n+2}{2}$ *implies* $diam(G) \leq 2$.
- $k \geq \frac{n+2}{2}$ *implies* $diam(G) \leq 2k - n + 2$.

Density Relative to A Clique

Another way of measuring the density is to express what fraction of all possible edges are contained in a graph, i.e., how close it is to a clique in terms of the number of edges.

Definition 6. *Let G be an n-vertex graph with m edges, and let $0 \leq \eta \leq 1$ be a real number. Then, G is said to be η-dense if*

$$\frac{m}{n(n-1)/2} \geq \eta.$$

Not surprisingly, it is **NP**-complete to decide whether an η-dense subgraph of a given size exists in an input graph. This follows from the results of the next subsection as a special case, due to the fact that an η-dense subgraph of a given size is a subgraph of a given size having at least a given number of edges.

A notable feature of η-density is that it is inherited by subgraphs, in a sense similar to the clique property. However, the property is not as strong as in the case of cliques: if in a clique we delete any vertex, then we get a clique again. In an η-dense graph, one can only prove that there is a vertex whose deletion preserves the η-density, but this may not hold for all vertices. Note that many other density measures do not have even this weaker inheritance property. Specifically, one can prove (see Kosub [19]):

Theorem 8. *Let $0 \leq \eta \leq 1$ be a real number. An η-dense n-vertex graph always contains an η-dense $(n-1)$-vertex subgraph.*

Constrained size Subgraphs with Maximum Number of Edges

Another reasonable concept of a densest subgraph, approximating a clique, is expressed by the following problem: a number k as input and we look for a k-vertex subgraph with the maximum number of edges. As it contains MAXCLIQUE as a special case, it is **NP**-hard. Interestingly, as proved by Asahiro, Hassin and Iwama [28], it remains **NP**-hard even if we are satisfied with a subgraph that has only $\Theta(k^{1+\epsilon})$ edges, where $0 < \epsilon < 1$ is any fixed constant. On the other hand, it can be approximated in polynomial time within approximation ratio $n^{1/4+\epsilon}$, for every $\epsilon > 0$, in time $n^{O(1/\epsilon)}$, as proved by Bhaskara et al. [29].

For special graph classes, stronger results are available. For example, polynomial-time constant factor approximations have been developed for many intersection graph classes (see Chen, Fleischer, and Li [30]). If the whole input graph is dense, which means that the minimum degree is at least δn for some constant $\delta > 0$, then the problem has a Polynomial-Time Approximation Scheme (PTAS), as proved by Arora, Karger, and Karpinski [31].

There are some inapproximability results about this problem, as well:

- There is a no polynomial-time approximation algorithm with an approximation factor of $n^{1/(\log\log n)^c}$ for some $c > 0$, if the so-called Exponential Time Hypothesis (ETH) is true, as proved by Manurangsi [32]. The ETH asserts that the 3-SATISFIABILITY problem is not solvable faster than exponential time in the worst case. This is a stronger assumption than **P** $\neq$ **NP**.
- Khot [33] proved that the problem does not have a Polynomial-Time Approximation Scheme (PTAS) for the general case, unless **NP**-complete problems can be solved in subexponential time with randomization. For dense input graphs, however, there is a PTAS, as mentioned above. Recall that dense means the minimum degree is at least δn for some constant $\delta > 0$.

Two interesting variants of the problem arise when instead of requiring exactly k vertices in the subgraph, we require at least k or at most k, where the density is interpreted by Definition 6. The first variant is abbreviated DalkS (Dense at least k-vertex Subgraph), the second is abbreviated DamkS (Dense at most k-vertex Subgraph). They behave quite differently, as shown by Chen et al. [14]. Specifically, DalkS is solvable in polynomial time, if k is bounded by a constant. Note that the constant bound is not on the size of the subgraph, it is on the *minimum* size k that the subgraph must have. Once having $\geq k$ vertices, the subgraph can be arbitrarily large. In the general case, the task is **NP**-hard. DamkS is also **NP**-hard, but can be approximated within a factor of 2.

3.2.3. Dense Common Subgraphs in Multiple Graphs

Thus far, we have been considering variants of the densest subgraph problem in a single graph. The task gets additional twists and hardness, if multiple graphs are considered in parallel.

A common framework is (see, e.g., Charikar, Naamad, and Wu [34] and Semertzidis, Pitoura, Terzi, and Tsaparas [35]) when, instead of single graph, we are given a *sequence* of graphs on the same vertex set, and are looking for a subgraph, referred to as *Densest Common Subgraph (DCS)*, that maximizes some aggregate measure of density over the graphs. The graph sequence may represent a temporal aspect, for example, a network that is changing in time. The individual graphs in the sequence are referred to as *frames*.

Some typical aggregate density measures are:

- DCS-MA, where the objective is to maximize the minimum over the frames of the average degree in the induced subgraph (induced by the node set selected for the subgraph).
- DCS-AM where the objective is to maximize the average over the frames of the minimum degree in the induced subgraph.
- DCS-MM, where the objective is to maximize the minimum over the frames of the minimum degree in the induced subgraph.
- DCS-AA, where the objective is to maximize the average over the frames of the average degree in the induced subgraph.

Various results are presented about these objectives in [34,35], for example, the first two objectives lead to **NP**-hard optimization, while the other two are solvable in polynomial time.

4. Some Related Results and Open Problems

4.1. Conditions that Force the Existence of a Dense Subgraph

Given that some of the dense subgraph types are hard to find or even to approximate in large graphs, it can be useful to have results that *guarantee* the existence of such subgraphs. The typical form of such theorems is this: if an n-vertex graph has at least a certain number of edges, then it is guaranteed to have a certain type of subgraph. Let us briefly review some examples below.

A famous and fundamental early result is Turán's theorem [36]:

Theorem 9. *Let G be an n-vertex graph with m edges. If $m > \frac{n^2}{2}\frac{k-2}{k-1}$, then G contains a k-clique.*

A generalization of Turán's theorem was proved by Dirac [37]:

Theorem 10. *Let G be an n-vertex graph with m edges. Let k, r be integers with $n \geq k + r$ and $0 \leq r \leq k + 2$. If $m > \frac{n^2}{2}\frac{k-2}{k-1}$, then G contains a subgraph on $k + r$ vertices, such that its average degree is $k + r - 1 - \frac{r}{k+r}$.*

It is worth looking at some special cases of Dirac's theorem. If $r = 0$, then the conclusion is that there is a k-vertex subgraph with average degree $k - 1$, which is exactly a k-clique. Therefore, Turán's theorem is included as a special case. If, for example, $k = r \leq n/2$, then we obtain the conclusion

that there is a subgraph on $2k$ nodes with average degree $2k - 3/2$. This is not a clique, but it is still quite close to that, since a $2k$-vertex graph with average degree $2k - 1$ would be a $2k$-clique. Thus, by relaxing the density slightly to $2k - 3/2$, we can double the size of the guaranteed dense subgraph.

Another generalization of Turán's theorem is a powerful result of Erdős and Stone [38]. It provides a sufficient condition, similar to Turán's theorem, to contain an *arbitrary* given subgraph H, rather than a k-clique. The role of k is taken over here by the *chromatic number* $\chi(H)$ of H, which is the minimum number of colors needed to assign a color to each vertex of H, such that neighboring nodes have different colors.

Theorem 11. *Let G be an n-vertex graph with m edges. Let H be an arbitrary graph with chromatic number $\chi(H) = k$. There is a sequence ϑ_n with $\lim_{n\to\infty} \vartheta_n = 0$ such that for each n-vertex graph G with m edges, if $m > \frac{n^2}{2}\left(\frac{k-2}{k-1} + \vartheta_n\right)$, then G contains a subgraph isomorphic to H.*

Turán's theorem and the Erdős–Stone theorem, both originated in the 1940s, became cornerstones of extremal graph theory.

There is also a result of similar spirit, due to Lick and White [8], that provides sufficient condition for the existence of a k-core (for definition, see Section 3.1.2):

Theorem 12. *Let G be an n-vertex graph with m edges, and let k be an integer with $1 \leq k \leq n-1$. If $m > (k-1)n - \frac{k(k-1)}{2} + 1$, then G has a non-empty k-core.*

4.2. Min-Max Theorems

There are many results in combinatorial optimization, as well as related areas, that are referred to as *min-max theorems*, also called *min-max characterization*. The general form of such theorems is outlined below.

Let $f(x)$ be a polynomial-time computable function that maps a bit-string x into a rational number. The optimization problem is this: What is the maximum value of $f(x)$ over n-bit strings x? We say that such a problem has a *min-max characterization*, if there is another polynomial-time computable function g, such that

$$\max_x f(x) = \min_y g(y)$$

holds. Here, x runs over all n-bit strings, and y runs over all m-bit strings; n and m may be different, but they are polynomially related.

Numerous natural and important optimization problems have such a min-max characterization (see, e.g., Schrijver [39]). Let us list a few examples, showing the theorems on which the characterizations are based in parenthesis: Linear Programming (LP Duality Theorem), Maximum Flow (Max Flow Min Cut Theorem), Maximum Bipartite Matching (Kőnig–Hall Theorem), Maximum Non-Bipartite Matching (Tutte's Theorem and Tutte–Berge formula), Maximum Disjoint Arborescences in directed graph (Edmond's Disjoint Branching Theorem), Maximum Spanning Tree Packing in undirected graph (Tutte's Tree Packing Theorem), Minimum Covering by Forests (Nash–Williams Theorem), Maximum Directed Cut Packing (Lucchesi–Younger Theorem), Max 2-Matroid Intersection (Matroid Intersection Theorem), Maximum Disjoint Paths (Menger's Theorem), Max Antichain in Partially Ordered Set (Dilworth's Theorem), and a number of others.

The significance of min-max theorems is two-fold:

- They provide non-trivial insight into the problem structure, which is often helpful in developing algorithms.
- They offer a way to *certify* the optimality of a solution. For example, if someone provides us a network flow in a large network, and claims that it is a maximum flow, how can this be certified without re-running the entire algorithm? We can easily check that the flow is *feasible*, by simply checking that it satisfies all the constraints. How do we make sure it is indeed *optimal*, i.e., there is

no flow with larger value? If we are also given a cut (a dual solution) with capacity equal to the flow value, then it indeed certifies the optimality, since no more flow can get through than what the bottleneck allows. The fact that such an optimality certificate always exists for this problem is not trivial, it follows from the Max Flow Min Cut Theorem.

Now, we turn to the question: Are there such min-max characterizations for some of the densest subgraph problems we have considered? We show three examples below.

Gabow and Westermann [40] proved a characterization for the density of the densest subgraph, with respect to the edge density (see Section 3.1.1):

Theorem 13 (Pseudo-arboricity). *Let us call a graph a* pseudo-forest *if every connected component in it has at most one cycle. Let G be a graph, and let $\kappa(G)$ denote the* minimum *number k, such that there are k edge-disjoint pseudo-forests that together cover all edges of G. This quantity is called the* pseudo-arboricity *of G. Then, the following holds:*

$$\kappa(G) = \left\lceil \max_S \frac{|E(S)|}{|V(S)|} \right\rceil,$$

where $\lceil . \rceil$ denotes the upper integer part of any number.

For example, if we want to convince somebody that a large graph has a subgraph with edge density 32.6, then we can simply exhibit such a subgraph (found by one of the known algorithms). However, how can we certify that such a subgraph does *not* exist? This can be done by exhibiting a covering of the graph edges with at most 32 disjoint pseudo-forests. It proves that $\kappa(G) \leq 32$, and by the above theorem it implies that G cannot contain a subgraph with edge density greater than 32.

Another min-max theorem was proved by Matula [17] about the connectivity of the most connected subgraph (see Section 3.1.8):

Theorem 14 (Slicing). *Let us call a family of edge cuts in a graph a* slicing, *if the union of these cuts contains all edges. The* width *of the slicing is the size of the largest cut in the family. Then, the minimum width of any slicing is equal to the maximum edge-connectivity of any subgraph.*

There is also a min-max theorem about the k-core. It is related to the so-called *coloring number* of the graph (not to be confused with the chromatic number), see Erdős and Hajnal [41]. The coloring number is the smallest number ℓ, such that the vertices can be ordered in a way that every vertex has at most $\ell - 1$ neighbors that precede it in the ordering. Such an order allows the coloring of the nodes with at most ℓ colors, by the simple greedy algorithm that visits the vertices in this ordering, and assigns to each node the first available color, which has not occurred among its preceding neighbors. Let us denote the coloring number of a graph G by $col(G)$.

Theorem 15 (Coloring number). *The largest value of k for which a graph G has a non-empty k-core is equal to $col(G) - 1$.*

4.3. Open Problems

In this section, we review several densest subgraph related problems that are still open (according to our best knowledge).

4.3.1. Density with P-computable Function of Degrees

In Section 3.1.4, we review a density measure that can incorporate more sophisticated functions of node degrees than just their sum, as in the edge density. For example, we see that the density measure

$$\frac{\sum_x d_S^2(x)}{|V(S)|}$$

can be optimized in polynomial time, as a special case of the **F**-density. This suggests that we may allow other functions of the degrees. Let $f(x)$ be a polynomial-time computable, real valued function, and consider the density measure

$$\rho(S, f) = \frac{\sum_x f(d_S(x))}{|V(S)|}.$$

Find the most general class of functions f for which the density measure $\rho(S, f)$ can be optimized in polynomial time.

4.3.2. Linear Combination of Polynomially Solvable Densities

Let $\rho'(S)$ and $\rho''(S)$ be two density measures that can be optimized in polynomial time. Consider the combined density measure

$$\rho(S) = \alpha\rho'(S) + \beta\rho''(S),$$

where α, β are constants. What conditions guarantee that $\rho(S)$ inherits the polynomial-time solvability? As an example, can we find a subgraph S in polynomial time, such that the edge density *plus* the edge-connectivity of S is maximized?

4.3.3. Min-Max Theorems for Other (Polynomially Solvable) Densities

In Section 4.2, we see min-max theorems for certain density measures. Of course, we can expect such results only for the polynomially solvable cases, since such a characterization for an **NP**-complete measure would imply **NP** = co-**NP**, which is highly unlikely.

It seems, however, that not all polynomially solvable cases have a "nice" min-max characterization. For example, we do not know such a characterization for the density measure

$$\frac{\sum_x d_S^2(x)}{|V(S)|},$$

even though we know it can be optimized polynomial time. It would be interesting to devise a min-max theorem for this, and other polynomially solvable cases.

4.3.4. Further Conditions that Force the Existence of A Dense Subgraph

In Section 4.1, we see that sometimes simple conditions on the number of edges can force the existence of certain types of dense subgraphs. It would be interesting to provide such conditions for other density measures, as well.

4.3.5. Maximum Clique in Random Graph

As shown in Section 3.2.1, the MAXCLIQUE problem is notoriously hard, it is unlikely to have even a reasonable approximation in the general case. On the other hand, there is more hope if the input graph is *random*.

For the sake of an example, let us consider the simplest random graph model, in which there are n vertices, and each edge is put in randomly and independently with probability $1/2$ (coin flip). This model is often denoted by $G(n, 1/2)$. It is known from the theory of random graphs (see, e.g., Bollobás [42]) that the size of a maximum clique in this graph is asymptotically $2\log_2 n$. How hard is it to *find* such a clique? It is unlikely to be **NP**-hard, because there are only $n^{O(\log n)}$ many subsets of this size, so we can do an exhaustive search in $n^{O(\log n)}$ time, which is still a sub-exponential algorithm. Even though it is unlikely to be **NP**-hard, no polynomial-time algorithm is known, either. In fact, about *half* of the optimum, i.e. $\log_2 n$, can be achieved by a simple polynomial-time algorithm, with high probability (see Grimmett and McDiarmid [43]). At the same time, for any fixed $\epsilon > 0$, no polynomial-time algorithm is known that would find a clique of size at least $(1 + \epsilon)\log_2 n$ with high probability.

5. Conclusions

We review density measures that play an important role in various applications. Some of these measures give rise to a densest subgraph concept that can be solved in polynomial time, while others lead to **NP**-hard optimization problems. However, even in the hard cases, various approximations are often available, although not always. This survey, as well as the presented open problems and accompanying results, shows that this is a vibrant, active area of research, which is still likely to generate many further results and publications.

Author Contributions: Writing—original draft, A.F.; writing—review and editing, Z.R.M.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Luce, R.D.; Perry, A. A method of matrix analysis of group structure. *Psychometrika* **1949**, *14*, 95–116. [CrossRef] [PubMed]
2. Gionis, A.; Tsourakakis, C.E. Dense Subgraph Discovery. KDD Tutorial. In Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'15), Sydney, Australia, 10–13 August 2015.
3. Picard, J.-C.; Queyranne, M. A Network Flow Solution to Some Nonlinear 0–1 Programming Problems with Application to Graph Theory. *Networks* **1982**, *12*, 141–159. [CrossRef]
4. Gallo, G.; Grigoriadis, M.D.; Tarjan, R.E. A Fast Parametric Maximum Flow Algorithm and Applications. *SIAM J. Comput.* **1989**, *18*, 30–55. [CrossRef]
5. Charikar, M. Greedy Approximation Algorithms for Finding Dense Components in a Graph. In *Approximation Algorithms for Combinatorial Optimization: Third International Workshop, APPROX 2000*; Springer: Berlin, Germany, 2000; pp. 84–95.
6. Dong, J.; Liu, Y. Determination of the Densest Subgraph. *J. Syst. Sci. Complex.* **2004**, *17*, 23–27.
7. Faragó, A. A General Tractable Density Concept for Graphs. *Math. Comput. Sci.* **2008**, *1*, 689-699. [CrossRef]
8. Lick, D.R.; White, A.T. *k*-Degenerate Graphs. *Can. J. Math.* **1970**, *22*, 1082–1096. [CrossRef]
9. Matula, D.W.; Beck, L.L. Smallest-Last Ordering and Clustering and Graph Coloring Algorithms. *J. ACM* **1983**, *30*, 417–427. [CrossRef]
10. Tatti, N.; Gionis, A. Density-Friendly Graph Decomposition. In Proceedings of the 24th International World Wide Web Conference (WWW'15), Florence, Italy, 18–22 May 2015; pp. 1089–1099.
11. Tsourakakis, C.E. The K-clique Densest Subgraph Problem. In Proceedings of the 24th International World Wide Web Conference (WWW'15), Florence, Italy, 18–22 May 2015; pp. 1122–1132.
12. Kawase, Y.; Miyauchi, A. The Densest Subgraph Problem with a Convex/Concave Size Function. *Algorithmica* **2018**, *80*, 3461–3480. [CrossRef]
13. Saha, B.; Hoch, A.; Khuller, S.; Raschid, L.; Zhang, X.-N. Dense Subgraphs With Restrictions and Applications to Gene Annotation Graphs. In *RECOMB 2010*; Berger, B., Ed.; Springer: Heidelberg, Germany, 2010; Volume 6044, pp. 456–472.
14. Chen, W.; Peng, L.; Wang, J.; Li, F.; Tang, M. Algorithms for the Densest Subgraph With at Least *k* Vertices and with a Specified Subset. In *Combinatorial Optimization and Applications*; Lu, Z., Kim, D., Wu, W., Li, W., Du, D.Z., Eds.; Springer: Berlin, Germany, 2015; Volume 9486, pp. 566–573.
15. Miyauchi, A.; Kakimura, N. Finding a Dense Subgraph with Sparse Cut. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM'18), Torino, Italy, 22–26 October 2018; pp. 547–556.
16. Nagamochi, H.; Ibaraki, T.; *Algorithmic Aspects of Graph Connectivity*; Cambridge University Press: Cambridge, UK, 2008.
17. Matula, D.W. The Cohesive Strength of Graphs. In *The Many Facets of Graph Theory*; Lecture Notes in Mathematics; Chartrand, G., Kapoor, S.F., Eds.; Springer: Berlin, Germany, 1969; Volume 110.
18. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W. H. Freeman and Co.: San Francisco, CA, USA, 1979.

19. Kosub, S. Local Density. In *Network Analysis—Methodological Foundations*; Brandes, U., Erlebach, T., Eds.; Springer: Berlin, Germany, 2005.
20. Robson, J.M. Algorithms for maximum independent sets. *J. Algorithms* **1986**, *7*, 425–440. [CrossRef]
21. Robson, J.M. Finding a Maximum Independent Set in Time $O(2^{n/4})$. Technical Report. 2001. Available online: http://www.labri.fr/perso/robson/mis/techrep.html (accessed on 10 June 2019).
22. Boppana, R.B.; Halldorsson, M.M. Approximating maximum independent sets by excluding subgraphs. *BIT Numer. Math.* **1992**, *32*, 180–196. [CrossRef]
23. Feige, U. Approximating Maximum Clique by Removing Subgraphs. *SIAM J. Discret. Math.* **2004**, *18*, 219–225. [CrossRef]
24. Zuckerman, D. Linear Degree Extractors and the Inapproximability of Max Clique and Chromatic Number. *Theory Comput.* **2007**, *3*, 103–128. [CrossRef]
25. ISGCI: Information System on Graph Classes and their Inclusions. Available online: http://www.graphclasses.org (accessed on 5 May 2019).
26. Tsukiyama, S.; Ide, M.; Ariyoshi, H.; Shirakawa, I. A New Algorithm for Generating all the Maximal Independent Sets. *SIAM J. Comput.* **1977**, *6*, 505–517. [CrossRef]
27. Johnson, D.S.; Papadimitriou, C.H.; Yannakakis, M. On generating all maximal independent sets. *Inf. Process. Lett.* **1988**, *27*, 119–123. [CrossRef]
28. Asahiro, Y.; Hassin, R.; Iwama, K. Complexity of Finding Dense Subgraphs. *Discret. Appl. Math.* **2002**, *121*, 15–26. [CrossRef]
29. Bhaskara, A.; Charikar, M.; Chlamtac, E.; Feige, U.; Vijayaraghavan, A. Detecting High Log-densities—An $O(n^{1/4})$ Approximation for Densest k-Subgraph. In Proceedings of the Annual ACM Symposium on Theory of Computing (STOC 2010), Cambridge, MA, USA, 6–8 June 2010; ACM: New York, NY, USA, 2010; pp. 201–210.
30. Chen, D.Z.; Fleischer, R.; Li, J. Densest k-Subgraph Approximation on Intersection Graphs. In *Approximation and Online Algorithms (WAOA 2010)*; Jansen, K., Solis-Oba, R., Eds.; Springer: Berlin, Germany, 2010; Volume 6534, pp. 84–93.
31. Arora, S.; Karger, D.; Karpinski, M. Polynomial Time Approximation Schemes for Dense Instances of NP-Hard Problems. *J. Comput. Syst. Sci.* **1999**, *58*, 193–210. [CrossRef]
32. Manurangsi, P. Almost-polynomial Ratio ETH-hardness of Approximating Densest k-Subgraph. In Proceedings of the 49th Annual ACM Symposium on Theory of Computing (STOC 2017), Montreal, PQ, Canada, 19–23 June 2017; pp. 954–961.
33. Khot, S. Ruling Out PTAS for Graph Min-Bisection, Densest Subgraph and Bipartite Clique. In Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04), Rome, Italy, 17–19 October 2004; pp. 136–145.
34. Charikar, M.; Naamad, Y.; Wu, J. On Finding Dense Common Subgraphs. *arXiv* **2018**, arXiv:1802.06361.
35. Semertzidis, K.; Pitoura, E.; Terzi, E.; Tsaparas, P. Finding lasting dense subgraphs. In *Data Mining and Knowledge Discovery*; Springer: Berlin, Germany, 2018; pp. 1–29.
36. Turán, P. On an Extremal Problem in Graph Theory. *Matematikai és Fizikai Lapok (Math. Phys. Lett.)* **1941**, *48*, 436–452.
37. Dirac, G.A. Extensions of Turáns Theorem on Graphs. *Acta Math. Acad. Sci. Hung.* **1963**, *14*, 417–422. [CrossRef]
38. Erdős, P.; Stone, A.H. On the Structure of Linear Graphs. *Bull. Am. Math. Soc.* **1946**, *52*, 1087–1091. [CrossRef]
39. Schrijver, A. Min-Max Results in Combinatorial Optimization. In *Mathematical Programming—The State of the Art*; Bachem, A., Korte, B., Grotschel, M., Eds.; Springer: Berlin, Germany, 1983.
40. Gabow, H.N.; Westermann, H.H. Forests, Frames, and Games: Algorithms for Matroid Sums and Applications. *Algorithmica* **1992**, *7*, 465–497. [CrossRef]
41. Erdős, P.; Hajnal, A. On Chromatic Number of Graphs and Set-Systems. *Acta Math. Hung.* **1966**, *17*, 61–99. [CrossRef]

42. Bollobás, B. *Random Graphs*; Cambridge University Press: Cambridge, UK, 2001.
43. Grimmett, G.; McDiarmid, C. On Colouring Random Graphs. *Math. Proc. Cam. Philos. Soc.* **1975**, *77*, 313–324. [CrossRef]