Computer Networks 52 (2008) 2831–2841

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Finding a dense-core in Jellyfish graphs

Mira Gonen *, Dana Ron [1], Udi Weinsberg, Avishai Wool

School of Electrical Engineering, Tel-Aviv University, Ramat Aviv, Israel

ARTICLE INFO

Article history:
Available online 13 June 2008

Keywords:
AS-graph
Dense-core
Sublinear algorithm

ABSTRACT

The connectivity of the Internet crucially depends on the relationships between thousands of Autonomous Systems (ASes) that exchange routing information using the Border Gateway Protocol (BGP). These relationships can be modeled as a graph, called the AS-graph, in which the vertices model the ASes, and the edges model the peering arrangements between the ASes. Based on topological studies, it is widely believed that the Internet graph contains a central dense-core: Informally, this is a small set of high-degree, tightly interconnected ASes that participate in a large fraction of end-to-end routes. Finding this dense-core is a very important practical task when analyzing the Internet's topology.

In this work we introduce a randomized *sublinear* algorithm that finds a dense-core of the AS-graph. We mathematically prove the correctness of our algorithm, bound the density of the core it returns, and analyze its running time. We also implemented our algorithm and tested it on real AS-graph data and on real undirected version of WWW network data. Our results show that the core discovered by our algorithm is nearly identical to the cores found by existing algorithms – at a fraction of the running time.

1. Introduction

1.1. Background and motivation

The connectivity of the Internet crucially depends on the relationships between thousands of Autonomous Systems (ASes) that exchange routing information using the Border Gateway Protocol (BGP). These relationships can be modeled as a graph, called the AS-graph, in which the vertices model the ASes, and the edges model the peering arrangements between the ASes.

Significant progress has been made in the study of the AS-graph's topology over the last few years. A great deal of effort has been spent measuring topological features of the Internet. Numerous research projects [16,1,15,26,36,37,11,6,5,25,31,27,9,8,29,35,33,34,22,10,30,32] have ventured to capture the Internet's topology. Based on these and other topological studies, it is widely believed that the Internet graph contains a central dense-core: Informally, this is a small set of high-degree, tightly interconnected ASes that participate in a large fraction of end-to-end routes. Finding this dense-core is a very important practical task when analyzing the Internet's topology.

There are several ways to define a dense-core precisely, and various corresponding algorithms and heuristics. In the next subsection, we briefly survey known definitions and algorithms, and shortly discuss their pros and cons. The goal of our work is to describe an algorithm that finds the dense-core (using a reasonable definition of a dense-core), is amenable to rigorous mathematical analysis, and is efficient, both asymptotically and when implemented and tested on real AS data, and on the undirected version of the WWW network data.

1.2. Defining a dense-core

An early conceptual model for the Internet topology was suggested by Tauro et al. [35]. This work seems to have coined the "Jellyfish" term. The authors argued that the Internet topology resembles a Jellyfish where the

* Corresponding author.
E-mail addresses: gonenmir@post.tau.ac.il (M. Gonen), danar@eng.tau.ac.il (D. Ron), udiw@eng.tau.ac.il (U. Weinsberg), yash@acm.org (A. Wool).

[1] Supported bu the Israel Science Foundation (Grant No. 89/05).

1389-1286/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2008.06.005

Internet core corresponds to the middle of the cap, which is surrounded by many "tentacles".

The simplest working definition of a dense-core is from Siganos et al. [32]: according to this work, a core is a clique of maximum size. Since the MaxClique problem is NP-hard and is even hard to approximate [23], the authors suggest a greedy algorithm, which we call GreedyMaxClique: Select the highest degree node as the first member of the core. Then, examine each node in decreasing degree order, and add to the core any node that neighbors *all* the nodes already in the core. This algorithm has a time complexity of $O(|E|)$ (where $|E|$ is the number of edges in the graph). On real AS data (with $n \approx 20,000$ and $|E| \approx 60,000$) the algorithm finds a clique of size 13. Since the graph is sparse (that is, $|E| = O(n)$), the algorithm works quite fast. However, the definition of the core as a clique is very restrictive, since it requires 100% edge density,[2] and there is no guarantee that the algorithm will indeed find even an approximately maximum clique. In this work we shall refer to such a clique as the *nucleus* of the AS-graph, to distinguish it from other definitions.

Carmi et al. [12,13] give a different definition for a dense-core. According to their definition, a k-dense-core is a *maximal* set of nodes with degrees $> k$, where the degree is measured in the subgraph induced by the core nodes. Alvarez-Hamelin et al. [3] use a similar k-core decomposition. Carmi et al. [12,13] described an algorithm to iteratively compute a k-core, which we refer to as the kCore algorithm. For a given minimal degree k, kCore repeatedly eliminates nodes with (residual) degrees $\leqslant k$, until no more nodes can be eliminated—and the remaining nodes form a k-core. On real AS-graph data, with $k = 30$, they get a core of about 100 nodes. The implementation of the kCore algorithm within the DIMES project [31] has a theoretical time complexity[3] of $O(n^2)$, and in practice it is significantly slower than the GreedyMaxClique algorithm of [32]. Note that even though the algorithm claims to find a "dense-core", it is really based on degrees and has a rather weak guarantee about the density of the resulting core: for a degree k, if the discovered core is C then the edge density is $> k/(|C| - 1)$. Furthermore, *a-priori* there is no guarantee on the size of the core that is found or on the discovered density: for a fixed degree k, one can construct an infinite family of connected graphs in which all nodes have degree $\geqslant k$ and the core density tends to 0.[4]

Subramanian et al. [33] suggested a 5-tier hierarchical layering of the AS-graph. Their dense-core – the top tier, is defined as a subset of ASes whose edge density is >50%. Their tiering agrees with the Jellyfish model of [35] in that they, implicitly, assume a single dense-core. They use a simple greedy algorithm for finding (their definition of) a dense-core. However, they report finding a dense-core of only 20 ASes. A similar approach was suggested by Ge et al. [20].

Feige et al. [17] consider the k-densest subgraph problem, which is defined as follows. Given a graph $G = (V, E)$, find a subgraph $H = (X, F)$ of G such that $|X| = k$ and $|F|$ is maximized. This problem is NP-hard. Feige et al. [17] describe an approximation algorithm that gives a ratio of $O(n^\delta)$ for some $\delta < 1/3$. For any particular value of k the greedy algorithm of Asahiro et al. [4] (which is similar to the kCore algorithm) gives the ratio $O(n/k)$. For some specific values of k there are algorithms that produce approximation ratios that are better than $O(n/k)$ [18,19]. Charikar [14] considers the related problem of finding a subgraph of maximum average degree. He shows that a simple (linear time) greedy algorithm, which is a variant of the kCore algorithm and the algorithm of Asahiro et al. [4], gives a factor-2 approximation. The proof is based on the relation between the greedy algorithm and the dual of the LP formulation of the problem. We note that in general, a subset of (approximately) maximum average degree might be quite different from the notion we are interested in of a relatively small, very dense subgraph. The example given in Footnote 4 illustrates this.

Sagie and Wool [30] suggested an approach that is based on dense k-subgraphs (*DkS*). They use parts of the *DkS* approximation algorithm of [17]. On a sampled AS-graph (based on BGP data) their algorithm found a dense-core of 43 ASes, with density 70%. The time complexity of their algorithm is rather high: $O(n^3)$. Bar et al. [5,6] use the same approach for finding a dense-core.

Against this backdrop of diverging definitions, our goal was to design an algorithm that (i) is not limited to finding a fully-connected clique, (ii) provides a precise density guarantee for the discovered core, (iii) is very efficient, both asymptotically and in practice, and (iv) is amenable to mathematical analysis.

1.3. Contributions

We chose to use a natural definition of a dense-core that focuses on the actual edge density: We define a dense-core as a set of vertices of size k with a given density α. Motivated by graph property-testing algorithms [21], our approach is to use randomized sampling techniques to find such a core. A related approach was applied in [28] to find large conjunctive clusters. However, intuitively, a dense-core in a general graph is a "local" phenomenon, so random sampling has a very low success probability (e.g., if the core is log-sized). Therefore, we restrict ourselves to the practically-interesting class of Jellyfish graphs: graphs that contain a dense-core – and this core is also well connected to other parts of the graph.

The extra structure provided by Jellyfish graphs is the basis for our main contribution: a *sublinear* randomized algorithm for finding a dense-core in the AS-graph. We rigorously prove the correctness of our algorithm, and the density of the dense-core produced by the algorithm under mild structural assumptions on the graph (assumptions that do hold for the real AS-graph).

We implemented our algorithm (JellyCore) and tested it extensively on AS-graph data collected by the DIMES project

[2] The density of a subgraph with k vertices is the fraction of the $k(k-1)/2$ possible edges that exist in the subgraph.

[3] A more careful implementation, using a bucket-based priority queue, would have time complexity $O(n \log n)$. We have not attempted to improve the DIMES implementation.

[4] For example, take a collection of m k-cliques and connect them via m additional edges. All nodes have a degree of k or $k+1$ so the core is the whole graph. As m grows the density vanishes.

[31]. We compared our results to those produced by the DIMES [31] implementation of kCore [12] and the GreedyMaxClique algorithm of Siganos et al. [32]. On the AS-graph our JellyCore algorithm finds a 60-80%-dense-core, which has a 90% overlap with the core reported by kCore – but JellyCore runs six times faster. Furthermore, we also define a *nucleus* within the dense-core as a subset of the highest degree vertices in the dense-core. The nucleus produced by our algorithm has an 80–90% overlap with the 13-node clique reported by GreedyMaxClique – i.e., we find a nucleus containing around 11 of 13 members in the clique.

Organization: In Section 2, we give definitions and notations. In Section 3, we describe and analyze a simple randomized algorithm (JellyCore) for finding the dense-core, which serves as a basis for our sublinear algorithm. In Section 4, we modify the JellyCore algorithm to a sublinear algorithm. In Section 5, we give an implementation of the JellyCore algorithm and compare it to the algorithms of Carmi et al. [12] and Siganos et al. [32]. We summarize our conclusions in Section 6.

2. Definitions and notations

Throughout the paper we consider sparse graphs $G = (V, E)$, i.e., $|E| = O(n)$, where $n = |V|$. For the purpose of time complexity analysis, we assume that for every vertex in the graph we know the degree (in $O(1)$ time). We start by some technical definitions leading up to the definition of the dense-core, and the family of Jellyfish graphs.

Definition 1. Closeness to a clique: Let C^k denote the k-vertex clique. Denote by $\text{dist}(G, C^k)$ the distance (as a fraction of $\binom{k}{2}$) between a graph G over k vertices and C^k. Namely, if $\text{dist}(G, C^k) = \epsilon$ then $\epsilon\binom{k}{2}$ edges should be added in order to make G into a clique. A graph G over k vertices is ϵ-close to being a clique if $\text{dist}(G, C^k) \leqslant \epsilon$.

Definition 2. (k, ϵ)-dense-core: consider a graph G. A subset of k vertices in the graph is a (k, ϵ)-*dense-core* if the subgraph induced by this set is ϵ-close to a clique.

Definition 3. Let C be a subset of vertices of a graph G. The d-*nucleus* of C, denoted by H, is the subset of vertices of C with degree (*not* induced degree) at least d.

For a set of vertices X, let $\Gamma(X)$ denote the set of vertices that neighbor at least one vertex in X, and let $\Gamma_\delta(X)$ denote the set of vertices that neighbor at least $(1-\delta)|X|$ vertices in X. We next introduce our main definition.

Definition 4. (k, d, c, ϵ)-Jellyfish subgraph: For integers k and d, and for $0 \leqslant \epsilon \leqslant 1$ (that may all be functions of n), and for a constant $c \geqslant 1$, a graph G contains a (k, d, c, ϵ)-Jellyfish subgraph if it contains a subset C of vertices, with $|C| = k$, that is a (k, ϵ)-dense-core, which has a non-empty d-nucleus H s.t. the following conditions hold:

1. For all $v \in C$, $v \in \Gamma_\epsilon(H)$.
2. For all but $\epsilon|\Gamma_{3\epsilon}(H)|$ vertices, if a vertex $v \in V$ is in $\Gamma_\epsilon(H)$ then $v \in \Gamma_\epsilon(C)$.
3. For all but $\epsilon|H|$ vertices in the graph, if $\deg(v) \geqslant d$ then $v \in H$.
4. $|\Gamma_{3\epsilon}(H)|/|C| \leqslant c$.

Intuitively, Item 1 of Definition 4 describes the fact that the vertices in C have many neighbors in H. Item 2 describes the fact that vertices that have many neighbors in H must have many neighbors in C too (so that the neighborhood relation to H is in a sense "representative" of the neighborhood relation to C). Item 3 describes the fact that most of the high-degree vertices in the graph are in H. Item 4 describes the fact that most vertices that neighbor most of H, are in C. Thus Items 1–4 describe the dense-core as a dense set of vertices that contains most of the very high-degree nodes in the graph, neighbors most of these high-degree nodes, and are almost all the vertices in the graph that neighbor most of these high-degree nodes. Fig. 1 shows an illustration of a Jellyfish graph.

Two notes are in place:

1. In Section 5, JellyCore is run for the values of k, d, c, ϵ for which these assumptions hold for the AS-graph according to [31].
2. Item 3 in Definition 4 will be relaxed in Section 3.1.

3. The JellyCore algorithm for finding a dense-core in Jellyfish graphs

In this section, we describe a randomized algorithm that, given a graph $G = (V, E)$ that contains a (k, d, c, ϵ)-Jellyfish subgraph, finds a $(k, (8 \cdot c + 1)\sqrt{\epsilon})$-dense-core $\widehat{C}$ and an approximation of the nucleus H. The set $\widehat{C}$ can be verified to be a $(k, (8 \cdot c + 1)\sqrt{\epsilon})$-dense-core. Thus if the graph does not contain a (k, d, c, ϵ)-Jellyfish subgraph then if the set returned by our algorithm is not a $(k, (8 \cdot c + 1)\sqrt{\epsilon})$-dense-core we can detect it. Our algorithm and its analysis take some ideas from the Approximate-Clique Finding Algorithm of [21] (which is designed for dense graphs).

The algorithm is given query access to the graph G, and takes as input: k (the requested dense-core size), d (the minimal degree for nodes in the nucleus), ϵ, and a sample size s.

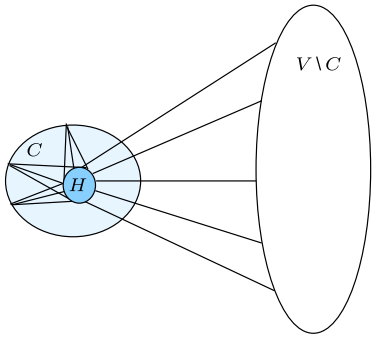

Fig. 1. An illustration of a Jellyfish graph.

Algorithm 1 (*The JellyCore algorithm for approximating C and H*).

1. Uniformly and independently at random select s vertices. Let S be the set of vertices selected.
2. Compute $\widehat{H} = \{v \in \Gamma(S) | \deg(v) \geqslant d\}$. If $\widehat{H} = \emptyset$ then *abort*.
3. Compute the set $\Gamma_{2\epsilon}(\widehat{H})$ of vertices that neighbor all but at most of $2\epsilon|\widehat{H}|$ vertices in $\widehat{H}$.
4. Order the vertices in $\Gamma_{2\epsilon}(\widehat{H})$ according to their degree in the subgraph induced by $\Gamma_{2\epsilon}(\widehat{H})$ (breaking ties arbitrarily). Let $\widehat{C}$ be the first k vertices according to this order.
5. Return $\widehat{C}, \widehat{H}$.

Our main result is the following:

Theorem 1. *Let $G = (V, E)$ be a sparse graph that contains a (k, d, c, ϵ)-Jellyfish subgraph. Then, for $s \geqslant c'(n/d)\ln(|H| + 2)$, where c' is a constant, with probability at least $1 - e^{-(c'-1)}$, Algorithm 1 finds a set $\widehat{C}$ of size $|\widehat{C}| = k$ that is $O(\sqrt{\epsilon})$ close to being a clique, and finds a set $\widehat{H}$ that is a superset of H s.t. $|\widehat{H}| \leqslant (1 + \epsilon)|H|$. The time complexity of Algorithm 1 is $O(n \log n)$.*

Intuitively, the algorithm works in graphs that contain (k, d, c, ϵ)-Jellyfish subgraphs since in such graphs it suffices to sample a small set of vertices and observe their neighbors. The set of the neighbors with degree at least d is close to a nucleus H. In addition, in graphs that contain (k, d, c, ϵ)-Jellyfish subgraphs each vertex in C neighbors most of the vertices in H, and there might be only few vertices outside C that neighbor most of the vertices in H. Therefore, by taking the vertices that neighbor most of the vertices in H we get an approximation of C. However, in general graphs, if we sample a small set of vertices, the set of their neighbors might be a small random subset, so we will not be able to get any approximation of C.

We prove Theorem 1 by proving several lemmas. We first state the following technical lemma which establishes that there cannot be too many numbers that are much smaller than their average:

Lemma 1. *Let $X = \{x_1, \ldots, x_n\}$ be a set of positive integers, and let $x_{\max} = \max\{x_1, \ldots, x_n\}$. Then for every r:*

$$\left|\left\{x_j \in X | x_j < \left[\frac{1}{n}\sum_{i=1}^{n} x_i - r\left(x_{\max} - \frac{1}{n}\sum_{i=1}^{n} x_i\right)\right]\right\}\right| < \frac{n}{r}. \quad (1)$$

Proof. Assume to the contrary that there are at least n/r elements in X for which Eq. (1) holds. Then

$$\sum_{j=1}^{n} x_j < \left[\frac{1}{n}\sum_{i=1}^{n} x_i - r\left(x_{\max} - \frac{1}{n}\sum_{i=1}^{n} x_i\right)\right] \cdot \frac{n}{r} + \left(n - \frac{n}{r}\right)x_{\max}. \quad (2)$$

Therefore

$$\sum_{j=1}^{n} x_j < \left(1 + \frac{1}{r}\right)\sum_{i=1}^{n} x_i - \frac{n}{r} \cdot x_{\max}. \quad (3)$$

This implies that

$$n \cdot x_{\max} < \sum_{j=1}^{n} x_j \quad (4)$$

a contradiction. □

Assume for now that we have access to a superset $\overline{U}$ of H that contains vertices with degree at least d. By Item 3 in Definition 4, it holds that $|\overline{U}| \leqslant (1 + \epsilon)|H|$. Then the next lemma shows that the (unknown) dense-core C is a subset of the 2ϵ-neighborhood of $\overline{U}$.

Lemma 2. $C \subseteq \Gamma_{2\epsilon}(\overline{U}) \subseteq \Gamma_{3\epsilon}(H)$.

Proof. According to the definition of $\Gamma_{2\epsilon}(\overline{U})$ it holds that each vertex in $\Gamma_{2\epsilon}(\overline{U})$ neighbors at least $(1 - 2\epsilon)|\overline{U}|$ vertices in $\overline{U}$. This implies that every vertex in $\Gamma_{2\epsilon}(\overline{U})$ neighbors at least $(1 - \frac{2\epsilon}{1-\epsilon})|H|$ vertices in H. Since it holds that $(1 - \frac{2\epsilon}{1-\epsilon}) > 1 - 3\epsilon$ we get that $\Gamma_{2\epsilon}(\overline{U}) \subseteq \Gamma_{3\epsilon}(H)$. In addition, each vertex in C neighbors at least $(1 - \epsilon)|H|$ vertices in H. Since $(1 - \epsilon)|H| \geqslant (1 - \epsilon)\frac{|\overline{U}|}{1+\epsilon} > (1 - \epsilon)(1 - \epsilon)|\overline{U}| > (1 - 2\epsilon)|\overline{U}|$, we get that $C \subseteq \Gamma_{2\epsilon}(\overline{U})$. □

We next state and prove our main lemma.

Lemma 3. *Suppose we order the vertices in $\Gamma_{2\epsilon}(\overline{U})$ according to their degree in the subgraph induced by $\Gamma_{2\epsilon}(\overline{U})$ (breaking ties arbitrarily). Let $\widehat{C}$ be the first k vertices according to this order. Then $\widehat{C}$ is $O(\sqrt{\epsilon})$ close to being a clique.*

Proof. Let $R = \{v \in \Gamma_{2\epsilon}(\overline{U}) \setminus C : |\Gamma(v) \cap C| \geqslant (1 - \epsilon)|C|\}$, and let $B = \Gamma_{2\epsilon}(\overline{U}) \setminus (C \cup R)$.

Fig. 2 shows a plot of the relations between H, C, R, and B. Since by Lemma 2 it holds that $\Gamma_{2\epsilon}(\overline{U}) \subseteq \Gamma_{3\epsilon}(H)$, according to Item 2 of Definition 4 it holds that

$$|B| < \epsilon|\Gamma_{3\epsilon}(H)|. \quad (5)$$

Using Item 4 of Definition 4 and the fact that $\Gamma_{2\epsilon}(\overline{U}) \subseteq \Gamma_{3\epsilon}(H)$ (again by Lemma 2) we get that

$$\frac{|R|}{|C| - 1} \leqslant \frac{|\Gamma_{2\epsilon}(\overline{U}) \setminus C|}{|C| - 1} \leqslant \frac{|\Gamma_{2\epsilon}(\overline{U})| - |C| + 1}{|C| - 1} \leqslant \frac{|\Gamma_{3\epsilon}(H)|}{|C| - 1} - 1 \leqslant \frac{c|C|}{|C| - 1} - 1 \leqslant 2c - 1. \quad (6)$$

Let the degree of vertex v in the subgraph induced by $\Gamma_{2\epsilon}(\overline{U})$ be denoted by $\deg_{\Gamma_{2\epsilon}(\overline{U})}(v)$. We have

$$\sum_{v \in C} \deg_{\Gamma_{2\epsilon}(\overline{U})}(v) \geqslant 2 \cdot |\{(u, v) | u, v \in C\}| + |\{(u, v) | u \in C, v \in R\}| \geqslant (1 - \epsilon)|C|(|C| - 1) + |R|(1 - \epsilon)|C| = |C|((1 - \epsilon)(|C| - 1) + |R|(1 - \epsilon)).$$

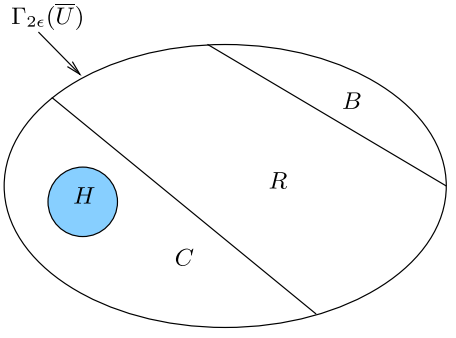

Fig. 2. An illustration of $\Gamma_{2\epsilon}(\overline{U})$.

Thus, the average value of $\deg_{\Gamma_{2\epsilon}(\overline{U})}(v)$ for $v \in C$ is at least $(1-\epsilon)(|C|-1)+|R|(1-\epsilon)$. On the other hand, since by Lemma 2 it holds that $C \subseteq \Gamma_{2\epsilon}(\overline{U})$, the maximum degree of $\deg_{\Gamma_{2\epsilon}(\overline{U})}(v)$ for $v \in C$ is bounded above by $|\Gamma_{2\epsilon}(\overline{U})|-1 = |C|-1+|R|+|B|$. Therefore, the difference between the maximum value and the average value of $\deg_{\Gamma_{2\epsilon}(\overline{U})}(v)$ for $v \in C$ is at most $|C|-1+|R|+|B| - ((1-\epsilon)(|C|-1)+|R|(1-\epsilon)) = \epsilon(|C|-1)+\epsilon|R|+|B|$. Using Lemma 1 we have that for every r

$$|\{v \in C | \deg_{\Gamma_{2\epsilon}(\overline{U})}(v) < (1-\epsilon)(|C|-1)+|R|(1-\epsilon) - r(\epsilon(|C|-1)+\epsilon|R|+|B|)\}| < |C|/r.$$

Thus at least $(1-1/r)|C|$ vertices in C have degree (in the subgraph induced by $\Gamma_{2\epsilon}(\overline{U})$) of at least

$$(1-\epsilon(r+1))(|C|-1)+|R|(1-\epsilon(r+1))-r|B|. \quad (7)$$

According to the definition of $\widehat{C}$ for at least $(1-1/r)|\widehat{C}|$ vertices in $\widehat{C}$ Eq. (7) holds. Let F be the set of vertices in $\widehat{C}$ for which Eq. (7) holds. Then, using Eqs. (6) and (5), for every $v \in F$,

$$\begin{aligned}
\deg_{\widehat{C}}(v) &\geqslant \deg_{\Gamma_{2\epsilon}(\overline{U})}(v) - |\Gamma_{2\epsilon}(\overline{U}) \setminus \widehat{C}| \\
&\geqslant (1-\epsilon(r+1))(|C|-1)+|R|(1-(r+1)\epsilon) \\
&\quad - r|B| - |\Gamma_{2\epsilon}(\overline{U}) \setminus \widehat{C}| \\
&= (1-\epsilon(r+1))(|C|-1)+|R|(1-(r+1)\epsilon) \\
&\quad - r|B| - |R| - |B| \\
&= (1-\epsilon(r+1))(|C|-1) - |R|(r+1)\epsilon - (r+1)|B| \\
&> (1-\epsilon(r+1))(|C|-1) - |R|(r+1)\epsilon \\
&\quad - (r+1)|\Gamma_{3\epsilon}(H)|\epsilon \\
&> (1-\epsilon(r+1))(|C|-1) - (2c-1)(|C|-1)(r+1)\epsilon \\
&\quad - (r+1)\epsilon 2c(|C|-1) \\
&= (|C|-1)(1-(r+1)4c\epsilon).
\end{aligned}$$

Summing up the degrees (in $\widehat{C}$) of all vertices in $\widehat{C}$, we obtain

$$\begin{aligned}
\sum_{v \in \widehat{C}} \deg_{\widehat{C}}(v) &= \sum_{v \in F} \deg_{\widehat{C}}(v) + \sum_{v \in \widehat{C} \setminus F} \deg_{\widehat{C}}(v) \\
&\geqslant \sum_{v \in F} (|C|-1)[1-4c(r+1)\epsilon] \\
&\geqslant |F|(|C|-1)[1-4c(r+1)\epsilon] \\
&\geqslant (1-1/r)|C|(|C|-1)[1-4c(r+1)\epsilon].
\end{aligned}$$

Setting $r = 1/\sqrt{\epsilon}$, we get

$$\begin{aligned}
\sum_{v \in \widehat{C}} \deg_{\widehat{C}}(v) &\geqslant (1-\sqrt{\epsilon})\left[1-4c(1/\sqrt{\epsilon}+1)\epsilon\right]|C|(|C|-1) \\
&> \left[1-(\sqrt{\epsilon}+4c(1/\sqrt{\epsilon}+1)\epsilon)\right]|C|(|C|-1) \\
&> \left[1-(8c+1)\sqrt{\epsilon}\right]|C|(|C|-1).
\end{aligned}$$

Therefore $\widehat{C}$ contains at least $[1-(8c+1)\sqrt{\epsilon}]|C|(|C|-1)/2$ edges. It follows that $\widehat{C}$ is $(8c+1)\sqrt{\epsilon}$-close to being a clique. □

Since we do not actually have access to a superset $\overline{U}$ of H that contains vertices with degree at least d, we sample the graph in order to get w.h.p. such a set.[5] Specifically, we select s vertices uniformly and independently, where s should be at least $c'(n/d)\ln(|H|+2)$ for a constant c', and let S denote the subset of sampled vertices. Let $\widehat{H} = \{v \in \Gamma(S) | \deg(v) \geqslant d\}$. Then

Lemma 4. *With probability at least $1-e^{-(c'-1)}$ it holds that $H \subseteq \widehat{H}$.*

Proof. Recall that S is constructed by s independent steps, where in each step we uniformly choose a vertex. We show that when using s trials, with probability at least $1-\frac{1}{e^{c'-1}}$ we cover all the vertices in H, i.e., for all $v \in H$ it holds that v is a neighbor of at least one vertex in S. According to Item 3 of Definition 4, $|\widehat{H}| \leqslant (1+\epsilon)|H|$. Fix some $v \in H$. The probability that $v \notin \widehat{H}$ is at most $(1-\frac{d}{n})^s < e^{-d\cdot s/n}$. Using the union bound we get that the probability that there exists some $v \in H$ that is not covered in any of the trials is at most $|H|e^{-d\cdot s/n}$, and since $s \geqslant c'(n/d)\ln(|H|+2)$ we get that $|H|e^{-d\cdot s/n} \leqslant \frac{1}{|H|^{c'-1}} \leqslant \frac{1}{e^{c'-1}}$. □

Proof of Theorem 1. The correctness of Algorithm 1 follows from Lemmas 3 and 4. It remains to compute the time complexity of the algorithm:

1. Steps 1 and 2: The most expensive operation is computing $\widehat{H}$. $\widehat{H}$ is computed by going over all the vertices in S, and adding the neighbors of each vertex with degree at least d to a list. (Thus a vertex can appear several times in the list). The time complexity is $\sum_{v \in S}\deg(v) \leqslant \min\{2|E|, |S|\cdot n\} = O(n)$.
2. Step 3: This step is preformed in the following manner. First the multiset $\Gamma(\widehat{H})$ is computed, and then $\Gamma_{2\epsilon}(\widehat{H})$ is computed.
   (a) $\Gamma(\widehat{H})$ is computed by going over all the vertices in $\widehat{H}$, and adding the neighbors of each vertex to a list. (Here too a vertex can appear several times in the list). The time complexity is $\sum_{v \in \widehat{H}}\deg(v) \leqslant \min\{2|E|, |\widehat{H}|\cdot n\} = O(n)$.
   (b) $\Gamma_{2\epsilon}(\widehat{H})$ is computed by the following algorithm: (i) Sort the vertices in the multiset $\Gamma(\widehat{H})$ according the names of the vertices. (ii) For each vertex in $\Gamma(\widehat{H})$ count the number of times it appears in $\Gamma(\widehat{H})$. If it appears at least $(1-2\epsilon)|\Gamma(\widehat{H})|$ times then add the vertex to $\Gamma_{2\epsilon}(\widehat{H})$. The time complexity is $|\Gamma(\widehat{H})|\log|\Gamma(\widehat{H})| = O(n\log n)$.
3. Step 4: $\widehat{C}$ is computed by first computing the degrees in $\Gamma_{2\epsilon}(\widehat{H})$ of the vertices in $\Gamma_{2\epsilon}(\widehat{H})$, and then sorting the vertices in $\Gamma_{2\epsilon}(\widehat{H})$ according to this degree. Computing

[5] We note that it is possible to search the graph for the vertices with degree at least d in linear time, which would not change (asymptotically) the running time of Algorithm 1. However, we shall need to perform random sampling in our sublinear algorithm, which is based on Algorithm 1, and hence we choose to introduce sampling at this stage. Furthermore, as we see in our implementation, in practice, we gain from using random sampling even when running Algorithm 1.

the degrees is upper bounded by $\sum_{v\in\Gamma_{2\epsilon}(\widehat{H})}\deg(v)=\min\{2|E|,n\cdot|\Gamma_{2\epsilon}(\widehat{H})|\}=O(n)$. Therefore, the time complexity of this step is upper bounded by $O(n)+|\Gamma_{2\epsilon}(\widehat{H})|\log(|\Gamma_{2\epsilon}(\widehat{H})|)\leqslant O(n)+O(k\log k)=O(n\log n)$.

Thus the time complexity of the algorithm is $O(n\log n)$. □

3.1. An algorithm for the case of relaxed assumptions

In this section, we show that we can relax Item 3 of Definition 4 to the case that ϵ is any constant (that might be larger than 1). Therefore we assume the following:

Definition 5. For all but $\bar{c}|H|$ vertices in the graph, if $\deg(v)\geqslant d$ then $v\in H$.

The algorithm presented is very similar to Algorithm 1, but since $\bar{c}$ might be large, we have to look for H by going over all the subsets of U that are of size H. Thus, if H is the only subset for which Definitions 4 and 5 hold, this algorithm *exactly* finds H w.h.p.

The algorithm is given query access to the graph G, and takes as input: k (the requested dense-core size), d (the minimal degree for nodes in the nucleus), $|H|$, the size of the nucleus, ϵ, and a sample size s.

Algorithm 2 (*An algorithm for approximating C* and *H*).

1. Uniformly and independently select s vertices. Let S be the set of vertices selected.
2. Compute $U=\{v\in\Gamma(S)|\deg(v)\geqslant d\}$. If $U=\emptyset$ then *abort*.
3. For each $U'\subseteq U$ of cardinality $|H|$, perform the following steps:
   (a) Compute $\Gamma_\epsilon(U')$, the set of vertices that neighbor all but at most $\epsilon|U'|$ of the vertices in U'.
   (b) Order the vertices in $\Gamma_\epsilon(U')$ according to their degree in the subgraph induced by $\Gamma_\epsilon(U')$ (breaking ties arbitrarily). Let $\widehat{C}(U')$ be the first k vertices according to this order.
4. Among all sets $\widehat{C}(U')$, let $\widehat{C}$ be the one that is closest to being a clique. Return $\widehat{C},U'$

We prove the following variation of Theorem 1:

Theorem 2. *Let $G=(V,E)$ be a sparse graph that contains a (k,d,c,ϵ)-Jellyfish subgraph. Then, for $s\geqslant c'(n/d)\ln(|H|+2)$, where c' is a constant, and $k=O(\log n)$, $|H|=O(\log k)$, with probability at least $1-o(1)$ Algorithm 2 finds a set $\widehat{C}$ of size $|\widehat{C}|=k$ that is $O(\sqrt{\epsilon})$ close to being a clique, and finds H. The time complexity of Algorithm 2 is $\widetilde{O}(n)$.*

Theorem 2 follows from several lemmas stated below.

Lemma 5. $C\subseteq\Gamma_\epsilon(H)$.

Lemma 6. *Assume we order the vertices in $\Gamma_\epsilon(H)$ according to their degree in the subgraph induced by $\Gamma_\epsilon(H)$ (breaking ties arbitrarily). Let $\widehat{C}$ be the first k vertices according to this order. Then $\widehat{C}$ is $O(\sqrt{\epsilon})$ close to being a clique.*

The proof of Lemma 5 follows directly from Item 1 of Definition 4. The proof of Lemma 6 is similar to the proof of Lemma 3.

Since we do not have access to such a set H we sample the graph in order to get H w.h.p. As in the previous section, we uniformly select $s=c'(n/d)\ln(|H|+2)$ vertices, where c' is a constant and let the resulting subset be denoted by S. As in the previous section, let $U=\{v\in\Gamma(S)|\deg(v)\geqslant d\}$. Then we get:

Lemma 7. *With probability at least $1-e^{-(c'-1)}$ it holds that $H\subseteq U$.*

Proof. Recall that S is constructed by s independent steps, where in each step we uniformly choose a vertex. We show that when using s trials, with probability at least $1-\frac{1}{e^{c'-1}}$ we cover all the vertices in H, i.e., for all $v\in H$ it holds that v is a neighbor of at least one vertex in S. According to Item 3 of Definition 4, $|U|\leqslant(1+\bar{c})|H|$. Fix some $v\in H$. The probability that $v\notin U$ is at most $\left(1-\frac{d}{n}\right)^s<e^{-d\cdot s/n}$. Using the union bound we get that the probability that there exists some $v\in H$ that is not covered in any of the trials is at most $|H|e^{-d\cdot s/n}$, and since $s\geqslant c'(n/d)\ln(|H|+2)$ we get that $|H|e^{-d\cdot s/n}\leqslant\frac{1}{|H|^{c'-1}}\leqslant\frac{1}{e^{c'-1}}$. □

Proof of Theorem 2. The correctness of Algorithm 2 follows from Lemmas 6 and 7. It remains to compute the time complexity of the algorithm (in a similar manner to the proof of Theorem 1):

1. Steps 1 and 2: $O(n)$.
2. Step 3: for every subset U' there are $O(n\log n)$ operations. Since the number of subsets U' is $\binom{|U'|}{|H|}\leqslant 2^{(1+\bar{c})|H|}$ we get that the overall time complexity of step 3 is $O(n\log n\cdot 2^{(1+\bar{c})|H|})$.
3. Step 4: finding the closeness to a clique of $\widehat{C}(U')$ takes $\binom{|\widehat{C}(U')|}{2}$. Thus, since $|\widehat{C}(U')|=k$, and there are $\binom{|U'|}{|H|}\widehat{C}(U')$'s, the time complexity of step 4 is upper bounded by

$$\begin{aligned}&\binom{|U'|}{|H|}\cdot\binom{|\widehat{C}(U')|}{2}\cdot(\min\{2|E|,|\Gamma_\epsilon(U')|\cdot n\}\\&\quad+|\Gamma_\epsilon(U')|\log|\Gamma_\epsilon(U')|)\\&=O(2^{(1+\bar{c})|H|})O(k^2)\min\{O(n),O(n\cdot k)\}+O(k\log k)\\&=O(n\cdot 2^{(1+\bar{c})|H|}k^2).\end{aligned}$$

Therefore, the time complexity of the algorithm is $O(n\log n\cdot k^2\cdot 2^{(1+\bar{c})|H|})$. For $k=O(\log n)$, $|H|=O(\log k)$ the time complexity is $\widetilde{O}(n)$. □

4. A sublinear algorithm for finding a dense-core in Jellyfish graphs

In this section, we modify the algorithm described in the previous section to get a sublinear algorithm that works under an additional assumption. For the sake of simplicity, we

continue using the term Jellyfish subgraph, where we only add an additional parameter to its definition. Specifically, we say that a graph $G = (V, E)$ contains a (k, d, d', c, ϵ)-Jellyfish subgraph if it contains a (k, d, c, ϵ)-Jellyfish subgraph as described in Definition 4, and there are at most $\epsilon|H|$ vertices in the graph with degree larger than d'.

The next claim follows directly from a simple counting argument.

Claim 8. *Let $G = (V, E)$ be a sparse graph, where $|E| \leqslant c''n$. Then for any choice of d, the graph G contains at most $d/2$ vertices with degree larger than $4c''n/d$.*

Let H' be a subset of H that contains only vertices with degree at most d'. By the definition of a (k, d, d', c, ϵ)-Jellyfish subgraph, it holds that

$$|H| \geqslant |H'| \geqslant |H| - \epsilon|H| = (1 - \epsilon)|H|.$$

The algorithm is given query access to the graph $G = (V, E)$, and takes as input: k (the requested dense-core size), d (the minimal degree for nodes in the nucleus), d' (the high-degree threshold), ϵ, c'' (where $|E| \leqslant c''n$) and a sample size s.

Algorithm 3 (*An algorithm for approximating C and H*).

1. Uniformly and independently at random select s vertices. Let S be the set of vertices selected.
2. Compute $S' = \{v \in S | \deg(v) \leqslant 4c''n/d\}$.
3. Compute $\widehat{H}' = \{v \in \Gamma(S') | d \leqslant \deg(v) \leqslant d'\}$. If $\widehat{H}' = \emptyset$ then *abort*.
4. Compute $\Gamma_{4\epsilon}(\widehat{H}')$ the set of vertices that neighbor all but at most $4\epsilon|\widehat{H}'|$ vertices in $\widehat{H}'$.
5. Compute $\widehat{C} = \{u \in \Gamma_{4\epsilon}(\widehat{H}') | \deg(u) \geqslant d\}$.
6. Order the vertices in $\Gamma_{4\epsilon}(\widehat{H}') \setminus \widehat{C}$ according to their degree in the subgraph induced by $\Gamma_{4\epsilon}(\widehat{H}')$ (breaking ties arbitrarily). Let C'' be the first $k - |\widehat{C}|$ vertices according to this order.
7. $\widehat{C} \leftarrow \widehat{C} \cup C''$.
8. Return $\widehat{C}, \widehat{H}'$.

Our main result is the following:

Theorem 3. *Let $G = (V, E)$ be a sparse graph that contains a $(k, \Omega(n^{1-\beta}), \mathrm{O}(n^{1-\beta/2}), c, \epsilon)$-Jellyfish subgraph. Then, for $s \geqslant c'(n/d)\ln(|H| + 2)$, where c' is a constant, with probability at least $1 - \mathrm{e}^{1-c'/2}$ Algorithm 3 finds a set $\widehat{C}$ of size $|\widehat{C}| = k$ that is $\mathrm{O}(\sqrt{\epsilon})$ close to being a clique, and finds a set $\widehat{H}'$ that is a superset of H' s.t. $|\widehat{H}'| \leqslant (1 + \epsilon)|H|$.[6] For $k = \mathrm{O}(\log n)$ and $\beta \leqslant 2/5$, the time complexity of Algorithm 3 is[7] $\tilde{\mathrm{O}}(n^{1-\beta/2})$.*

Assume first that we have an access to a superset U' of H' that contains vertices with degree at least d and at most d'. Then $|U'| \leqslant (1 + \epsilon)|H| \leqslant (1 + \epsilon)\frac{1}{1-\epsilon}|H'| \leqslant (1 + \epsilon)^2|H'|$.

Lemma 9. $C \subseteq \Gamma_{4\epsilon}(U') \subseteq \Gamma_{5\epsilon}(H)$.

Proof. According to the definition of $\Gamma_{4\epsilon}(U')$ it holds that each vertex in $\Gamma_{4\epsilon}(U')$ neighbors at least $(1 - 4\epsilon)|U'|$ vertices in U'. This implies that every vertex in $\Gamma_{4\epsilon}(U')$ neighbors at least $(1 - \frac{4\epsilon}{1-\epsilon})|H'|$ vertices in H'. Since it holds that $(1 - \frac{4\epsilon}{1-\epsilon}) > 1 - 5\epsilon$ we get that $\Gamma_{4\epsilon}(U') \subseteq \Gamma_{5\epsilon}(H)$. In addition, each vertex in C neighbors at least $(1 - \epsilon)|H|$ vertices in H. This implies that each vertex in C neighbors at least $(1 - \frac{\epsilon}{1-\epsilon})|H'|$ vertices in H'. Since $1 - \frac{\epsilon}{1-\epsilon} \geqslant 1 - 2\epsilon$, we get that every vertex in C neighbors at least $(1 - 2\epsilon)|H'|$ vertices in $H\prime$. Moreover, $(1 - 2\epsilon)|H'| \geqslant (1 - 2\epsilon)\frac{|U'|}{(1+\epsilon)^2} > (1 - 2\epsilon)(1 - \epsilon)^2|U'| > (1 - 4\epsilon)|U'|$. Thus $C \subseteq \Gamma_{4\epsilon}(U')$. □

We obtain the following lemma in the same manner as Lemma 3:

Lemma 10. *Let C'' be the set of vertices with degree at least d in $\Gamma_{4\epsilon}(U')$. Assume we order the vertices in $\Gamma_{4\epsilon}(U') \setminus C''$ according to their degree in the subgraph induced by $\Gamma_{4\epsilon}(U')$ (breaking ties arbitrarily). Let $\widehat{C}$ be the first $k - |C''|$ vertices according to this order. Then $C'' \cup \widehat{C}$ is $\mathrm{O}(\sqrt{\epsilon})$ close to being a clique.*

Since we do not have access to such a set U' we sample the graph in order to get w.h.p. a superset of H', denoted $\widehat{H}'$, that contains vertices with degree at least d and at most d'. We uniformly and independently at random select $s = c'(n/d)\ln(|H| + 2)$ vertices, where c' is a constant. Let S be the set of vertices selected, and let $S' = \{v \in S | \deg(v) \leqslant 4c''n/d\}$, and $\widehat{H}' = \{v \in \Gamma(S') | d \leqslant \deg(v) \leqslant d'\}$. Then

Lemma 11. *With probability at least $1 - \mathrm{e}^{1-c'/2}$ it holds that $H' \subseteq \widehat{H}'$.*

Proof. Recall that S is constructed by s independent steps, where in each step we uniformly choose a vertex. We show that when using s trials, with probability at least $1 - \frac{1}{\mathrm{e}^{c'/2-1}}$ we cover all the vertices in H, i.e., for all $v \in H$ it holds that v is a neighbor of at least one vertex in S. Fix some $v \in H'$. Then by Claim 8 the probability that $v \notin U$ is at most $(1 - \frac{d/2}{n})^s < \mathrm{e}^{-d \cdot s/(2n)}$. Using the union bound we get that the probability that there exists some $v \in H'$ that is not covered in any of the trials is at most $|H'|\mathrm{e}^{-d \cdot s/(2n)} \leqslant |H|^{1-c'/2} \leqslant \mathrm{e}^{1-c'/2}$. □

Proof of Theorem 3. The correctness of Algorithm 3 follows from Lemmas 10 and 11. It remains to compute the time complexity of the algorithm:

1. Steps 1, 2 and 3: the most expensive operation is computing $\widehat{H}'$. $\widehat{H}'$ is computed by going over all the vertices in S', and adding the neighbors of each vertex with degree at least d and at most d' to a list. (Thus a vertex can appear several times in the list). The time complexity is $\sum_{v \in S'} \deg(v) \leqslant \min\{2|E|, |S'| \cdot 4c''n/d\} = O(\min\{n, (n/d)^2 \log|H|\}) = O(n^{2\beta} \cdot \log k) = O(n^{1-\beta/2} \log k)$ (recall that $\beta \leqslant 2/5$).
2. Step 4: This step is preformed in the following manner: first the multiset $\Gamma(\widehat{H}')$ is computed, and then $\Gamma_{4\epsilon}(\widehat{H}')$ is computed.
   (a) $\Gamma(\widehat{H}')$ is computed by going over all the vertices in $\widehat{H}'$, and adding the neighbors of each such vertex to a list. (Thus a vertex can appear several times in the list). The time complexity is $\sum_{v \in \widehat{H}'} \deg(v) \leqslant \min\{2|E|, |\widehat{H}'| \cdot d'\} = O(\min\{n, |H| \cdot d'\}) = O(n^{1-\beta/2} \cdot k)$.

[6] Recall that $|H'| \geqslant (1 - \epsilon)|H|$, so Algorithm 3 indeed approximates H.

[7] The notation $\tilde{\mathrm{O}}(g(k))$ for a function g of a parameter k means $\mathrm{O}(g(k) \cdot \mathrm{polylog}(g(k)))$ where $\mathrm{polylog}(g(k)) = \log^c(g(k))$ for some constant c.

(b) $\Gamma_{4\epsilon}(\widehat{H}')$ is computed by the following algorithm: (i) Sort the vertices in the multiset $\Gamma(\widehat{H}')$ according the names of the vertices. (ii) For each vertex in $\Gamma(\widehat{H}')$ count the number of times it appears in $\Gamma(\widehat{H}')$. If it appears at least $(1-4\epsilon)|\widehat{H}'|$ times then add the vertex to $\Gamma_{4\epsilon}(\widehat{H}')$. The time complexity is $|\Gamma(\widehat{H}')|\log|\Gamma(\widehat{H}')|$. $|\Gamma(\widehat{H}')| = \sum_{v\in\widehat{H}'}\deg(v) \leqslant \min\{2|E|, |\widehat{H}'|\cdot d'\} = O(\min\{n, n^{1-\beta/2}\cdot k\})$. Thus the time complexity is $O(\min\{n\log n, n^{1-\beta/2}\cdot k\log(n\cdot k)\})$.

3. Step 5: $\widehat{C}$ is computed by going over $\Gamma_{4\epsilon}(\widehat{H}')$. The time complexity is $|\Gamma_{4\epsilon}(\widehat{H}')| = O(k)$.
4. Step 6: C'' is computed by computing the degrees in $\Gamma_{4\epsilon}(\widehat{H}')$ of the vertices in $\Gamma_{4\epsilon}(\widehat{H}')\setminus\widehat{C}$, and sorting the vertices in $\Gamma_{4\epsilon}(\widehat{H}')\setminus\widehat{C}$ according to these degrees. The time complexity of computing the degrees is upper bounded by

$$\sum_{v\in\Gamma_{4\epsilon}(\widehat{H}')\setminus\widehat{C}}\deg(v) \leqslant \min\{2|E|, |\Gamma_{4\epsilon}(\widehat{H}')|\cdot d\} = O(\min\{n, n^{1-\beta}k\}).$$

The time complexity of this step is upper bounded by

$$O(\min\{n, n^{1-\beta}k\}) + |\Gamma_{4\epsilon}(\widehat{H}')|\log(|\Gamma_{4\epsilon}(\widehat{H}')|) \leqslant O(\min\{n, n^{1-\beta}k\}) + O(k\log k).$$

Thus the time complexity of the algorithm is $O(\min\{n\log n, n^{1-\beta/2}\cdot k\log(n\cdot k)\})$. For $k=\log n$ the time complexity is $\widetilde{O}(n^{1-\beta/2})$. □

5. Implementation

To demonstrate the usefulness of our algorithms beyond their theoretical contribution, we conducted a performance evaluation of our algorithm in comparison with the GreedyMaxClique algorithm of Siganos et al. [32] and the kCore algorithm of Carmi et al. [12] on real AS-graph data.

For our own algorithm we implemented the basic Algorithm 1 of Section 3. We did not implement the sublinear algorithm of Section 4. The AS-graph contains only a handful of very high-degree vertices, so the main assumption of Section 4 holds anyway. This means that the refinements of the sublinear algorithm, which ensure that we do not process too many such vertices, would not bring significant gains. Moreover, the basic JellyCore algorithm gave us excellent running times (see below), so we opted for simplicity and ease of programming. We did not use any special data structures in the JellyCore implementation beyond generic graph data structures.

For the kCore algorithm we adopted the DIMES [31] implementation. This implementation uses generic graph data structures, and as a result it has a theoretical time complexity of $O(n^2)$. It is possible to implement kCore using more sophisticated data structures that are tuned to the specific needs of the algorithm. One possible data structure could be an array of buckets organized by node degree, with direct pointers that allow efficient (constant-time) relocation of a node from bucket to bucket. With such a data structure kCore would have an $O(n\log n)$ theoretical time complexity. However, it is unclear whether the additional sophistication will indeed result in faster run times on the graph sizes we care about. We have not attempted to introduce better data structures into kCore.

All three algorithms were implemented in Java, using Sun's Java 5, and the open source library JUNG [24] (Java Universal Network/Graph Framework). We ran the algorithms on a 3 GHz 4× multiprocessor Intel Xeon server with 4 GB RAM, running RedHat Linux kernel 2.6.9.

We tested the algorithms on AS-graphs constructed from data collected by the DIMES project [31]. DIMES is a large-scale distributed measurements effort that measures and tracks the evolution of the Internet from hundreds of different view-points, and provides detailed Internet topology graphs. We merged AS-graphs from consecutive weeks starting from the first week of 2006 until reaching a total of 64 weeks in February 2007. This resulted in AS-graphs that have a vertex count ranging from 11,000 to around 21,000 ASes.

All three algorithms accept the Internet AS-graph as an input. The kCore algorithm used a degree of 29 (i.e., it produced a core in which the minimal residual node degree is 30). The parameters for our JellyCore Algorithm 1 were set

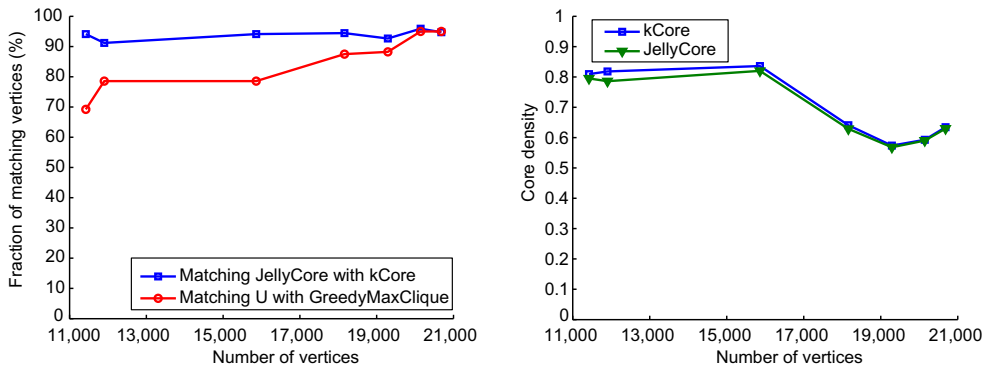

Fig. 3. Left: percentage of matching vertices between JellyCore and kCore, and percentage of matching vertices between of $U(=\widehat{H})$ and GreedyMaxClique, for increasing graph sizes. Right: core density.

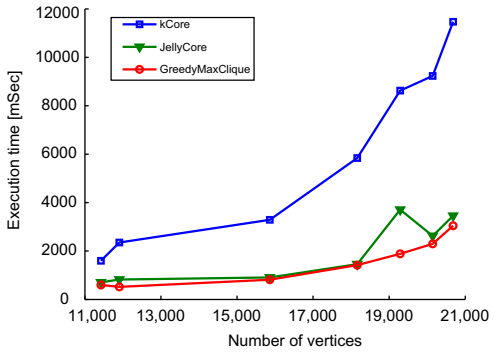

Fig. 4. Executions times.

Table 1
Empirical results

	JellyCore	kCore	GreedyMaxClique
Size	1357	1357	6
Density	0.22	0.22	1
Running time (s)	421	1857	431

as follows: Given the number of vertices n, we used a minimal nucleus degree of $d = n^{0.7}$, which gave $675 < d < 1100$ for our values of n. We picked $\epsilon = 0.1$ since we knew from earlier work that the AS-graph contains a clique of 10–13 vertices – a smaller value of ϵ would have been essentially meaningless. The sample size s was calculated as follows: $s = 10 \cdot n^{0.3} \cdot \ln(3 \log(5 \log n))$ (this gave values $473 < s < 577$ for our values of n). To allow a fair comparison with the kCore algorithm, we set the required dense-core size k to be the exact core size returned by the kCore algorithm in each run ($67 < k < 91$ in all cases).

Since the JellyCore algorithm is randomized, we ran it 10 times on each input graph, each time with independent random samples. Each point plotted in the figures represents the average of these 10 runs.

5.1. Accuracy of the JellyCore algorithm

Fig. 3 (left) shows the percentage of matching vertices of JellyCore and kCore. In other words, if kCore returned a core Z and Jellycore returned a core J then Fig. 3(left) shows $100 \cdot |J \cap Z|/|Z|$. We can see that in all cases, between 92% and 95% of the core J returned by JellyCore is also in Z. Thus the results of JellyCore and kCore are very similar on the AS-graph.

The figure also shows the percentage of matching vertices between the clique Q returned by GreedyMaxClique and the nucleus $\widehat{H}$ (here denoted by U). We can see that $\widehat{H}$ contains between 68% and 94% of the vertices of Q – and that this percentage improves as the number of vertices grows. Furthermore, we found by inspection that the JellyCore's J always completely includes the GreedyMaxClique Q.

Fig. 3(right) shows the density of the cores returned by JellyCore and of kCore as a function of the number of vertices of the graph. We can see that both densities are almost identical, particularly for $n \geqslant 18,000$ vertices. The density of GreedyMaxClique is obviously 1, by the algorithm definition.

We can conclude that the practical results of the JellyCore algorithm, on the real AS-graph, agree extremely well with the results of both kCore and GreedyMaxClique.

5.2. Execution times

Fig. 4 shows that the running times of JellyCore and GreedyMaxClique are almost identical, and that kCore is indeed slower: Jellycore runs about six times faster than kCore on the largest AS-graphs. Moreover, the running time of kCore increases substantially as the number of vertices in the graph grows, while the growth in the running times of JellyCore and GreedyMaxClique is relatively minor. Note that the running time of kCore may improve with better data structures – although it will surely remain slower than the much simpler GreedyMaxClique.

Therefore, we can see that the JellyCore algorithm produces cores that are very similar to those kCore – at a fraction of the running time. In addition, JellyCore returns "for free" the nucleus $\widehat{H}$, which is essentially the clique Q discovered by GreedyMaxClique.

5.3. Additional networks

To demonstrate the usefulness of our algorithms beyond the AS-graph, we conducted a performance evaluation of our algorithm in comparison with the GreedyMaxClique algorithm of Siganos et al. [32] and the kCore algorithm of Carmi et al. [12] on the undirected version of the WWW network within the nd.edu domain [7].

For our own algorithm we again implemented the basic Algorithm 1 of Section 3. All three algorithms were implemented in Java, using Sun's Java 5, using the open source library JUNG [24] (Java Universal Network/Graph Framework). We ran the algorithms on a 3 GHz 4× multiprocessor Intel Xeon server with 4 GB RAM, running RedHat Linux kernel 2.6.9.

We tested the algorithms on the undirected version of the WWW network within the nd.edu domain constructed from data collected by [7]. This network contains 325,682 vertices and 1,496,999 edges. Its average degree is 9.19. Its degree distribution follows a power-law with an exponent of 2.1 for the in-degree, and of 2.45 for the out-degree [2].

All three algorithms accept the above undirected version of the WWW network as an input. The kCore algorithm used a degree of 306 (i.e., it produced a core in which the minimal residual node degree is 307). The parameters for our JellyCore Algorithm 1 were set as follows: Given the number of vertices n, we used a minimal nucleus degree of $d = n^{0.7}$, which gave $d = 7227$ for our values of n. We picked $\epsilon = 0.1$. The sample size s was calculated as follows: $s = 10 \cdot n^{0.3} \cdot \ln(3 \log(5 \log n))$ (this gave value $s = 1340$ for our values of n). To allow a fair comparison with the kCore algorithm, we set the required dense-core size k to be the exact core size returned by the kCore algorithm ($k = 1357$).

Table 1 gives the running times of JellyCore, kCore, and GreedyMaxClique, the size of the returned core of all three algorithms, and the returned core's density of all three algorithms.

In addition, 98% of the vertices in the core returned by kCore are also in the core returned by JellyCore, five of six vertices in the core returned by GreedyMaxClique are also in the core returned by JellyCore, and three of six vertices in the core returned by GreedyMaxClique are also in the core returned by kCore. There are two vertices that are in the intersection of all cores.

Therefore, we can see that for the undirected version of the WWW network as well, the JellyCore algorithm produces cores that are very similar to those kCore – at a fraction of the running time. In addition, JellyCore returns "for free" the nucleus $\widehat{H}$, which is essentially the clique Q discovered by GreedyMaxClique.

6. Conclusions

In this work we presented first a simple algorithm (JellyCore), and then a *sublinear* algorithm, for approximating the dense-core of a Jellyfish graph. We mathematically proved the correctness of our algorithms, under mild assumptions that hold for the AS-graph. In our analysis we bounded the density of the cores our algorithms return, and analyzed their running time.

We also implemented our JellyCore algorithm and tested it on real AS-graph data, and on the undirected version of the WWW network within the nd.edu domain. Our results show that the dense-core returned by JellyCore is very similar to the kCore of Carmi et al. [12], at a fraction of the running time, and the improvement is more prominent as the number of vertices increases. In addition, as a side effect JellyCore also approximates the clique returned by GreedyMaxClique of Siganos et al. [32].

Therefore, we have demonstrated that our randomized approach provides both a theoretically successful algorithm (with a rigorous asymptotic analysis of the discovered density and success probability) – and a successful practical algorithm.

References

[1] R. Albert, A.-L. Barabási, Topology of evolving networks: local events and universality, Physical Review Letters 85 (24) (2000) 5234–5237.
[2] R. Albert, H. Jeong, A.-L. Barabási, Diameter of the World Wide Web, Nature 401 (1999) 130.
[3] I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, A. Vespignani, Large scale networks fingerprinting and visualization using the *k*-core decomposition, in: Proceedings of the Neural Information Processing Systems, August 2005.
[4] Y. Asahiro, K. Iwama, H. Tamaki, T. Tokuyama, Greedily finding a dense subgraph, Journal of Algorithms 34 (2000) 203–221.
[5] S. Bar, M. Gonen, A. Wool, An incremental super-linear preferential Internet topology model, in: Proceedings of the 5th Annual Passive and Active Measurement Workshop (PAM), LNCS 3015, Antibes Juan-les-Pins, France, April 2004, Springer-Verlag, pp. 53–62.
[6] S. Bar, M. Gonen, A. Wool, A geographic directed preferential Internet topology model, Computer Networks 51 (14) (2007) 4174–4188.
[7] A. Barabási and Z. Toroczkai, World-Wide-Web network within nd.edu domain, 1999, <http://www.www.nd.edu/networks/resources.htm>.
[8] P. Barford, A. Bestavros, J. Byers, M. Crovella, On the marginal utility of network topology measurements, in: Proceedings of the ACM SIGCOMM, 2001.
[9] G. Bianconi, A.L. Barabási, Competition and multiscaling in evolving networks, Europhysics Letters 54 (4) (2001) 436–442.
[10] R.X. Brunet, I.M. Sokolov, Evolving networks with disadvantaged long-range connections, Physical Review E 66 (026118) (2002).
[11] T. Bu, D. Towsley, On distinguishing between Internet power-law generators, in: Proceedings of the IEEE INFOCOM'02, New York, NY, USA, April 2002.
[12] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, E. Shir, Medusa – new model of Internet topology using k-shell decomposition, Technical Report, 2006. Available from: arXiv:cond-mat/0601240v1.
[13] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, E. Shir, A model of internet topology using k-shell decomposition, Proceedings of the National Academy of Sciences USA (PNAS) 104 (27) (2007) 11150–11154.
[14] M. Charikar, Greedy approximation algorithms for finding dense components in graphs, in: Proceedings of the APPROX, 2000.
[15] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. Shenker, W. Willinger, The origin of power laws in Internet topologies revisited, in: Proceedings of the IEEE INFOCOM'02, New York, NY, USA, April 2002.
[16] C. Faloutsos, M. Faloutsos, P. Faloutsos, On power-law relationships of the Internet topology, in: Proceeding of the ACM SIGCOMM'99, August 1999, pp. 251–260.
[17] U. Feige, G. Kortsarz, D. Peleg, The dense *k*-subgraph problem, Algorithmica 29 (3) (2001) 410–421.
[18] U. Feige, M. Langberg, Approximation algorithms for maximization problems arising in graph partitioning, Journal of Algorithms 41 (2001) 174–211.
[19] U. Feige, M. Seltser, On the densest *k*-subgraph problem. Technical report, Department of Applied Mathematics and Computer Science, The Weizmann Institute, Rehovot, 1997.
[20] Z. Ge, D.R. Figueiredo, S. Jaiswal, L. Gao, On the hierarchical structure of the logical Internet graph, in: SPIE ITCOM, August 2001.
[21] O. Goldreich, S. Goldwasser, D. Ron, Property testing and its connections to learning and approximation, Journal of the ACM 45 (1998) 653–750.
[22] R. Govindan, H. Tangmunarunki, Heuristics for Internet map discovery, in: Proceedings of the IEEE INFOCOM'00, Tel-Aviv, Israel, March 2000, pp. 1371–1380.
[23] J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, Acta Mathematica 182 (1999) 105–142.
[24] JUNG – the java universal network/graph framework, 2007. <http://jung.sourceforge.net/>.
[25] P.L. Krapivsky, G.J. Rodgers, S. Render, Degree distributions of growing networks, Physical Review Letters 86 (2001) 5401.
[26] A. Lakhina, J.W. Byers, M. Crovella, P. Xie, Sampling biases in IP topology measurements, in: Proceedings of the IEEE INFOCOM'03, 2003.
[27] X. Li, G. Chen, A local-world evolving network model, Physica A 328 (2003) 274–286.
[28] N. Mishra, D. Ron, R. Swaminathan, A new conceptual clustering framework, Machine Learning 56 (2004) 115–151.
[29] H. Reittu, I. Norros, On the power law random graph model of the Internet, Performance Evaluation 55 (January) (2004).
[30] G. Sagie, A. Wool, A clustering approach for exploring the Internet structure, in: Proceedings of the 23rd IEEE Convention of Electrical and Electronics Engineers in Israel (IEEEI), September 2004.
[31] Y. Shavitt, E. Shir, DIMES: Let the Internet measure itself, in: Proceedings of the ACM SIGCOMM, 2005, pp. 71–74.
[32] G. Siganos, S.L. Tauro, M. Faloutsos, Jellyfish: a conceptual model for the as Internet topology, Journal of Communications and Networks (2006).
[33] L. Subramanian, S. Agarwal, J. Rexford, R.H. Katz, Characterizing the Internet hierarchy from multiple vantage points, in: Proceedings of the IEEE INFOCOM'02, New York, NY, USA, April 2002.
[34] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, W. Willinger, Network topology generators: degree based vs. structural, in: Proceedings of the ACM SIGCOMM, 2002.
[35] L. Tauro, C. Palmer, G. Siganos, M. Faloutsos, A simple conceptual model for Internet topology, in: IEEE Global Internet, San Antonio, TX, November 2001.
[36] W. Willinger, R. Govindan, S. Jamin, V. Paxson, S. Shenker, Scaling phenomena in the Internet: critically examining criticality, Proceedings of the National Academy of Sciences of the United States of America 99 (February) (2002) 2573–2580.
[37] J. Winick, S. Jamin, Inet-3.0: Internet topology generator, Technical Report UM-CSE-TR-456-02, Department of EECS, University of Michigan, 2002.

Mira Gonen is a PhD candidate in the School of Electrical Engineering at Tel Aviv University, Israel. She received her B.A. from the Hebrew University, and her M.Sc. from the Open University. Her expected PhD graduation time is October 2008. Mira won the Intel Prize for excellent students, the Netvision Prize for excellent students, and a Doctoral Fellowship for Women in Science. Her research interests include randomized approximation algorithms, Internet topology and communication networks, and game theoretic aspects of the Internet.

Dana Ron is an Associate Professor in the School of Electrical Engineering at Tel Aviv University, Israel. She received her B.A., M.Sc., and Ph.D. from the Hebrew University in 1987, 1989, and 1995, respectively. She was an NSF postdoctoral fellow at MIT between 1995 and 1997, and a Bunting Fellow (ONR Science Scholar) at MIT and Radcliffe during 1997/8. She joined the faculty of Tel Aviv University in 1998. Her research interest include randomized approximation algorithms and in particular sublinear algorithms.

Udi Weinsberg is a PhD candidate in the School of Electrical Engineering at Tel Aviv University, Israel. He received his B.A. from the Technion, Israel Institute of Technology in 2001 and his M.Sc. from Tel-Aviv University in 2007. His research interest include Internet routing and topology analysis, complex networks and distributed large-scale graphs.

Avishai Wool received a B.Sc. (Cum Laude) in Mathematics and Computer Science from Tel Aviv University, Israel, in 1989. He received an M.Sc. and Ph.D. in Computer Science from the Weizmann Institute of Science, Israel, in 1993 and 1997, respectively. He then spent four years as a Member of Technical Staff at Bell Laboratories, Murray Hill, NJ, USA. In 2000 he co-founded Lumeta corporation, a startup company specializing in network security, and its successor, Algorithmic Security Inc. He is currently an Associate Professor at the School of Electrical Engineering, Tel Aviv University, Israel, where he has been since 2002.

He is the creator of the Firewall Analyzer. He is an associate editor of the ACM Transactions on Information and System Security. He has served on the program committee of the leading IEEE and ACM conferences on computer and network security. He is a senior member of IEEE, and a member the ACM and USENIX. His research interests include firewall technology, secure storage, computer, network and wireless security, smartcard-based systems, and the structure of the Internet topology.