

Finding a Dense-Core in Jellyfish Graphs

Mira Gonen, Dana Ron*, Udi Weinsberg, and Avishai Wool

Tel-Aviv University, Ramat Aviv, Israel
gonenmir@post.tau.ac.il, danar@eng.tau.ac.il,
udiw@eng.tau.ac.il, yash@acm.org

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. 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 [14, 1, 13, 25, 35, 36, 9, 5, 4, 24, 30, 26, 7, 6, 28, 34, 32, 33, 21, 8, 29, 31] 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

* This work was supported by the Israel Science Foundation (grant number 89/05).

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.

1.2 Defining a Dense-Core

An early conceptual model for the Internet topology was suggested by Tauro et al. [34]. This work seems to have coined the “jellyfish” term. The authors argued that the Internet topology resembles a jellyfish where the 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. [31]: 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 [22], 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 density¹, 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. [10, 11] 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. [2] use a similar k -core decomposition. Carmi et al. [10, 11] 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 $\leq k$, 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 kCore algorithm has a theoretical time complexity² of $O(n^2)$, and in practice it is significantly slower than the GreedyMaxClique algorithm of [31]. 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 $\geq k$ and the core density tends to 0.³

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

² A more careful implementation, using a bucket-based priority queue, gives complexity $O(n \log n)$.

³ E.g., 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.

Subramanian et al. [32] 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 [34] 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. [18].

Feige et al. [15] 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. [15] 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. [3] (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 $O(n/k)$ [16, 17]. Charikar [12] 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. [3], 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 3 illustrates this.

Sagie and Wool [29] suggested an approach that is based on dense k -subgraphs (DkS). They use parts of the DkS approximation algorithm of [15]. 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. [4, 5] 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 [19], our approach is to use randomized sampling techniques to find such a core. A related approach was applied in [27] 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 [30]. We also implemented the kCore algorithm of Carmi et al. [10], and the GreedyMaxClique algorithm of Siganos et al. [31]. 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 6 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 the 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. [10] and Siganos et al. [31]. 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 $\text{dist}(G, C^k) \leq \epsilon$.

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 all but at most $\delta|X|$ vertices in X . We next introduce our main definition.

Definition 4. (k, d, c, ϵ) -Jellyfish subgraph: For integers k and d , and for $0 \leq \epsilon \leq 1$ (that may all be functions of n), and for a constant $c \geq 1$, 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 neighbors at least $(1 - \epsilon)|H|$ vertices in H ,
2. For all but $\epsilon|\Gamma_{3\epsilon}(H)|$ vertices, if a vertex $v \in V$ neighbors at least $(1 - \epsilon)|H|$ vertices in H then v has at least $(1 - \epsilon)|C|$ neighbors in C .

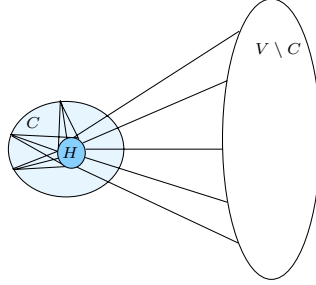


Fig. 1. An illustration of a Jellyfish Graph

3. For all but $\epsilon|H|$ vertices in the graph, if $\deg(v) \geq d$ then $v \in H$.
4. $|\Gamma_{3\epsilon}(H)|/|C| \leq c$.

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 to 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. Figure 1 shows an illustration of a Jellyfish graph.

Two notes are in place:

1. These assumptions hold for the AS-graph according to [30] for the values of k, d, c, ϵ we use.
2. Item 3 in Definition 4 will be relaxed in the full version of the paper [20] for $|H| = O(\log k)$ so that in this item ϵ will be any constant (that might be larger than 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 \hat{C} and an approximation of the nucleus H . Our algorithm and its analysis take some ideas from the Approximate-Clique Finding Algorithm of [19] (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 .

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 $\widehat{H} = \{v \in \Gamma(S) \mid \deg(v) \geq d\}$. 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 $s \geq c'(n/d) \ln(|H| + 2)$, 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. $|\widehat{H}| \leq (1 + \epsilon)|H|$. 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 won't be able to get any approximation of C .

We prove Theorem 1 by proving several lemmas. The lemmas and their proofs appear in the full version of the paper [20].

Assume for now that we have access to a superset \overline{U} of H that contains vertices with degree at least d .

We next state our main lemma.

Lemma 1. *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.*

Since we don't 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.⁴

⁴ 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.

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 $\widehat{H} = \{v \in \Gamma(S) \mid \deg(v) \geq d\}$. Then

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

Proof of Theorem 1. The correctness of Algorithm 1 follows from Lemmas 1 and 2. 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 $\sum_{v \in S} \deg(v) \leq \min\{2|E|, |S| \cdot n\} = O(n)$.
2. Step 3: This step is performed 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 $\sum_{v \in \widehat{H}} \deg(v) \leq \min\{2|E|, |\widehat{H}| \cdot n\} = O(n)$.
 - (b) $\Gamma_{2\epsilon}(\widehat{H})$ is computed by the following algorithm: (i) Sort the vertices in the multiset $\Gamma(\widehat{H})$ according to 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 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 $O(n) + |\Gamma_{2\epsilon}(\widehat{H})| \log(|\Gamma_{2\epsilon}(\widehat{H})|) \leq O(n) + O(k \log k) = O(n \log n)$.

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

4 A Sublinear Algorithm

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 3. *Let $G = (V, E)$ be a sparse graph, where $|E| \leq c''n$. 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

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

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 $|E| \leq c'n$) and a sample size s .

Algorithm 2. (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 $S' = \{v \in S \mid \deg(v) \leq 4c'n/d\}$.*
3. *Compute $\widehat{H}' = \{v \in \Gamma(S') \mid d \leq \deg(v) \leq d'\}$. 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 $\widehat{C} = \{u \in \Gamma_{4\epsilon}(\widehat{H}') \mid \deg(u) \geq d\}$.*
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 2. *Let $G = (V, E)$ be a sparse graph that contains a $(k, \Omega(n^{1-\beta}), O(n^{1-\beta/2}), c, \epsilon)$ -Jellyfish subgraph. Then, for $s \geq c'(n/d) \ln(|H| + 2)$, where c' is a constant, with probability at least $1 - e^{1-c'/2}$ Algorithm 2 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. $|\widehat{H}'| \leq (1 + \epsilon)|H|$.⁵ For $k = O(\log n)$ and $\beta \leq 2/5$, the time complexity of Algorithm 2 is $\tilde{O}(n^{1-\beta/2})$.⁶*

The proof of Theorem 2 appears in the full version of the paper [20].

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. [31] and the kCore algorithm of Carmi et al. [10] 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

⁵ Recall that $|H'| \geq (1 - \epsilon)|H|$, so Algorithm 2 indeed approximates H .

⁶ The notation $\tilde{O}(g(k))$ for a function g of a parameter k means $O(g(k) \cdot \text{polylog}(g(k)))$ where $\text{polylog}(g(k)) = \log^c(g(k))$ for some constant c .

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.

All three algorithms were implemented in Java, using Sun’s Java 5, using the open source library JUNG [23] (Java Universal Network/Graph Framework). We ran the algorithms on a 3GHz 4x multiprocessor Intel Xeon server with 4GB RAM, running RedHat Linux kernel 2.6.9.

We tested the algorithms on AS graphs constructed from data collected by the DIMES project [30]. 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 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

Figure 2 (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 Figure 2 (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 \hat{H} (here denoted by U). We can see that \hat{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 .

Figure 2 (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

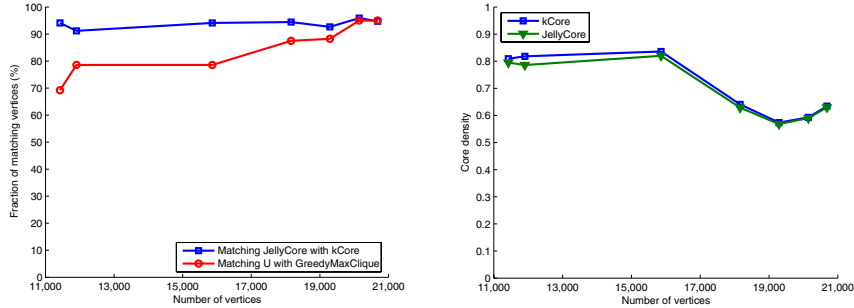


Fig. 2. Left: Percentage of matching vertices between JellyCore and kCore, and percentage of matching vertices between of $U(= \hat{H})$ and GreedyMaxClique, for increasing graph sizes. Right: Core Density.

almost identical, particularly for $n \geq 18,000$ 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.

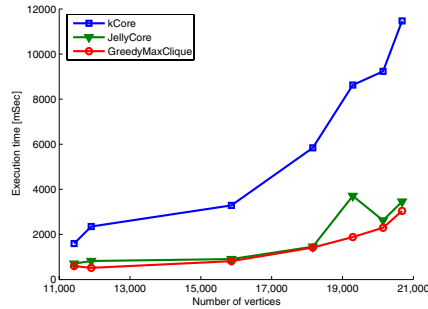


Fig. 3. Executions times

5.2 Execution Times

Figure 3 shows that the running times of JellyCore and GreedyMaxClique are almost identical, and that kCore is indeed slower: Jellycore runs about 6 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.

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 \hat{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. Our results show that the dense-core returned by JellyCore is very similar to the kCore of Carmi et al. [10], 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. [31].

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. Albert, R., Barabási, A.-L.: Topology of evolving networks: Local events and universality. *Physical Review Letters* 85(24), 5234–5237 (2000)
2. Alvarez-Hamelin, I., Dall’Asta, L., Barrat, A., Vespignani, A.: Large scale networks fingerprinting and visualization using the k-core decomposition. *Proc. Neural Information Processing Systems* (August 2005)
3. Asahiro, Y., Iwama, K., Tamaki, H., Tokuyama, T.: Greedily finding a dense subgraph. *Journal of Algorithms* 34, 203–221 (2000)
4. Bar, S., Gonen, M., Wool, A.: An incremental super-linear preferential Internet topology model. In: Barakat, C., Pratt, I. (eds.) *PAM 2004*. LNCS, vol. 3015, pp. 53–62. Springer, Heidelberg (2004)
5. Bar, S., Gonen, M., Wool, A.: A geographic directed preferential Internet topology model. *Computer Networks* 51(14), 4174–4188 (2007)
6. Barford, P., Bestavros, A., Byers, J., Crovella, M.: On the marginal utility of network topology measurements. In: *Proc. ACM SIGCOMM* (2001)
7. Bianconi, G., Barabási, A.L.: Competition and multiscaling in evolving networks. *Europhysics Letters* 54(4), 436–442 (2001)
8. Brunet, R.X., Sokolov, I.M.: Evolving networks with disadvantaged long-range connections. *Physical Review E* 66(026118) (2002)
9. Bu, T., Towsley, D.: On distinguishing between Internet power-law generators. In: *Proc. IEEE INFOCOM 2002*, New-York (April 2002)
10. Carmi, S., Havlin, S., Kirkpatrick, S., Shavitt, Y., Shir, E.: Medusa - new model of Internet topology using k-shell decomposition. Technical Report arXiv:cond-mat/0601240v1 (2006)
11. Carmi, S., Havlin, S., Kirkpatrick, S., Shavitt, Y., Shir, E.: A model of internet topology using k-shell decomposition. *PNAS* 2007. *Proceedings of the National Academy of Sciences, USA* 104(27), 11150–11154 (July 3, 2007)
12. Charikar, M.: Greedy approximation algorithms for finding dense components in graphs. In: *Proc. APPROX* (2000)
13. Chen, Q., Chang, H., Govindan, R., Jamin, S., Shenker, S., Willinger, W.: The origin of power laws in Internet topologies revisited. In: *Proc. IEEE INFOCOM 2002*, New-York (April 2002)

14. Faloutsos, C., Faloutsos, M., Faloutsos, P.: On power-law relationships of the Internet topology. In: Proc. of ACM SIGCOMM 1999, pp. 251–260 (August 1999)
15. Feige, U., Kortsarz, G., Peleg, D.: The dense k -subgraph problem. *Algorithmica* 29(3), 410–421 (2001)
16. Feige, U., Langberg, M.: Approximation algorithms for maximization problems arising in graph partitioning. *Journal of Algorithms* 41, 174–211 (2001)
17. Feige, U., Seltser, M.: On the densest k -subgraph problem. Technical report, Department of Applied Mathematics and Computer Science, The Weizmann Institute, Rehovot (1997)
18. Ge, Z., Figueiredo, D.R., Jaiswal, S., Gao, L.: On the hierarchical structure of the logical Internet graph. In: SPIE ITCOM (August 2001)
19. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connections to learning and approximation. *J. ACM* 45, 653–750 (1998)
20. Gonen, M., Ron, D., Weinsberg, U., Wool, A.: Finding a dense-core in jellyfish graphs. Technical report, School of Electrical Engineering, Tel-Aviv University (2007)
21. Govindan, R., Tangmunarunki, H.: Heuristics for Internet map discovery. In: Proc. IEEE INFOCOM 2000, Tel-Aviv, Israel, pp. 1371–1380 (March 2000)
22. Håstad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica* 182, 105–142 (1999)
23. JUNG - the java universal network/graph framework (2007), <http://jung.sourceforge.net/>
24. Krapivsky, P.L., Rodgers, G.J., Render, S.: Degree distributions of growing networks. *Physical Review Letters* 86(5401) (2001)
25. Lakhina, A., Byers, J.W., Crovella, M., Xie, P.: Sampling biases in IP topology measurements. In: Proc. IEEE INFOCOM 2003 (2003)
26. Li, X., Chen, G.: A local-world evolving network model. *Physica A* 328, 274–286 (2003)
27. Mishra, N., Ron, D., Swaminathan, R.: A new conceptual clustering framework. *Machine Learning* 56, 115–151 (2004)
28. Reittu, H., Norros, I.: On the power law random graph model of the Internet. *Performance Evaluation* 55 (January 2004)
29. Sagie, G., Wool, A.: A clustering approach for exploring the Internet structure. In: Proc. 23rd IEEE Convention of Electrical & Electronics Engineers in Israel (IEEEI) (2004)
30. Shavitt, Y., Shir, E.: DIMES: Let the Internet measure itself. In: Proc. ACM SIGCOMM, pp. 71–74 (2005)
31. Siganos, G., Tauro, S.L., Faloutsos, M.: Jellyfish: A conceptual model for the as Internet topology. *Journal of Communications and Networks* (2006)
32. Subramanian, L., Agarwal, S., Rexford, J., Katz, R.H.: Characterizing the Internet hierarchy from multiple vantage points. In: Proc. IEEE INFOCOM 2002, New-York (April 2002)
33. Tangmunarunkit, H., Govindan, R., Jamin, S., Shenker, S., Willinger, W.: Network topology generators: Degree based vs. structural. In: Proc. ACM SIGCOMM (2002)
34. Tauro, L., Palmer, C., Siganos, G., Faloutsos, M.: A simple conceptual model for Internet topology. In: IEEE Global Internet, San Antonio, TX (November 2001)
35. Willinger, W., Govindan, R., Jamin, S., Paxson, V., Shenker, S.: Scaling phenomena in the Internet: Critically examining criticality. *Proceedings of the National Academy of Sciences of the United States of America* 99, 2573–2580 (February 2002)
36. Winick, J., Jamin, S.: Inet-3.0: Internet topology generator. Technical Report UM-CSE-TR-456-02, Department of EECS, University of Michigan (2002)